STOCHASTIC RESOURCE CONTROL IN HETEROGENEOUS WIRELESS NETWORKS

by

Amin Farbod

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

Copyright © 2012 by Amin Farbod
Abstract

Stochastic Resource Control in Heterogeneous Wireless Networks

Amin Farbod
Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto
2012

In the near future, demand for Heterogeneous Wireless Networking (HWN) is expected to increase. HWNs are formed by integration of different communication technologies, for example the integration of wireless LAN and cellular networks, to support mobile users. QoS provisioning in these networks is a challenging issue given the diversity in wireless technologies and the existence of mobile users with different communication requirements. In this thesis, we consider optimal resource planning and dynamic resource management for HWNs.

In the first part of this thesis, we examine the optimal deployment of such networks. We propose a mobility-aware network planning optimization in which the objective is to minimize the rate of upward vertical handovers while maximizing the total number of users accommodated by the network. The optimal placement of Access Points (AP) with respect to these two objectives is formulated as an integer programming problem. Our results show that considering the mobility pattern in the planning phase of network deployment can significantly improve infrastructure performance.

In the second part, we investigate optimal admission control policies employed in maintaining QoS in HWNs. Here we consider two cases: integration of cellular overlay with a single WLAN AP, and integration with a WLAN mesh network. A decision theoretic framework for the problem is derived using a dynamic programming formulation.

In the case of single WLAN AP and cellular overlay, we prove that for this two-tier wireless network architecture, the optimal policy has a two-dimensional threshold structure. Further-
more, this structural result is used to design two computationally efficient algorithms, Structured Value Iteration and Structured Update Value Iteration. These algorithms can be used to determine the optimal policy in terms of thresholds. Although the first one is closer in its operation to the conventional Value Iteration algorithm, the second one has a significantly lower complexity.

In the second case where the underlay is a complex WLAN mesh network, we develop a Partially Observable Markov-Modulated Poisson Process (PO-MMPP) traffic model to characterize the overflow traffic from the underlaying mesh to the overlay. This model captures the burstiness of the overflow traffic under the imperfect observability of the mesh network states. Then, by modeling the overlay network as a controlled PO-MMPP/M/C/C queueing system and obtaining structured decision theoretic results, it is shown that the optimal control policy for this class of HWNs can be characterized as monotonic threshold curves. Moreover, these results are used to design a computationally efficient algorithm to determine the optimal policy in terms of thresholds.

Extensive numerical observations suggest that, in both cases and for all practical parameter sets, the algorithms converge to the overall optimal policy. Additionally, numerical results show that the proposed algorithms are efficient in terms of time-complexity and in achieving optimal performance by significantly reducing the probability of dropped and blocked calls.
TO MY PARENTS AND MY WIFE
Acknowledgements

I would like to express my gratitude to all those who supported me while I was working on my thesis. First and foremost, I am deeply grateful to my supervisor, Professor Ben Liang, for his detailed and insightful comments, and for his great support throughout this work. It was an honor to work under his supervision, and in addition to his invaluable technical help, I benefited from the very positive atmosphere that he had established in his research group. His broad understanding of telecommunications research, and his encouraging and personal guidance were instrumental in leading me to completing this work.

I am also extremely grateful to the members of my supervisory committee, Professor Alberto Leon-Garcia and Professor Baochun Li, whose constructive comments helped to improve this work, and to enhance the way it was presented. I am also grateful for their invaluable feedback on the direction of my work.

I would like to take this opportunity to thank administrative staff and other members of the Electrical and Computer Engineering department for their invaluable services and help throughout the course of my graduate studies at the University of Toronto.

Completing a doctoral thesis research is an extensive undertaking requiring sacrifices and broad support from family. It would have certainly been impossible to bring this work to fruition without the support of my parents and my wife. My parents, Touran and Shahin, ignited the desire to learn, and an inner and deep appreciation for science and knowledge that shaped my life more than I can express, and motivated me to complete this work. Finally, it was my wife, Nasrin, who stood by me with endless love and understanding while I was focused on my research.
# Contents

1 Introduction ................................................................. 1
   1.1 Motivations ......................................................... 3
   1.2 Contributions ...................................................... 4
   1.3 Thesis Outline ....................................................... 6

2 Related Work .................................................................. 7
   2.1 QoS and CAC in Heterogenous Wireless Networks ................. 7
   2.2 MDP Optimization of CAC for Wireless Networks ................. 10
   2.3 Structural Result for CAC ............................................ 12
   2.4 Overflow Modeling .................................................... 14

3 Backgrounds ................................................................... 18
   3.1 Dynamic Programming Framework ................................. 18
   3.2 Discrete Event Simulation .......................................... 22

4 Optimal Deployment of HWNs ........................................... 26
   4.1 Overview .............................................................. 26
   4.2 Background .......................................................... 28
   4.3 Assumptions and System Model .................................... 29
   4.4 Mobility and Equilibrium state ..................................... 30
   4.5 Optimization Problem Formulation ............................... 32
4.6 Simulation Results ........................................ 34
  4.6.1 Simulation Setup .................................. 34
  4.6.2 Effect of Initial Deployment ....................... 35
  4.6.3 Effect of Parameters Selection .................... 37
4.7 Chapter Summary ....................................... 39
4.8 Nomenclature .......................................... 40

5 Overlay and IEEE 802.11 Integration ............ 42
  5.1 Overview ............................................. 42
  5.2 Network Model ....................................... 44
  5.3 Optimal CAC Policy ................................ 46
    5.3.1 Optimality of Threshold-Based Policy ............ 47
    5.3.2 CAC Algorithm .................................. 49
    5.3.3 Finding Policy $\pi$ ............................... 51
  5.4 Efficient Computational Algorithms ................ 52
    5.4.1 Conventional VI Algorithm ....................... 52
    5.4.2 SVI Algorithm .................................. 55
    5.4.3 SUVI Algorithm ................................ 58
  5.5 Numerical Results .................................... 58
    5.5.1 The Optimal Policy ............................... 59
    5.5.2 Convergence Speed .............................. 60
    5.5.3 Optimal Policy vs. Complete Sharing (CS) ........ 62
    5.5.4 QoS Performance .................................. 66
  5.6 Chapter Summary ..................................... 68
  5.7 Nomenclature ........................................ 69

6 Overlay and Wireless Mesh Network Integration .... 71
  6.1 Overview ............................................ 71
Chapter 1

Introduction

Existing wireless communication technologies can generally be classified by their service model into two types: Local services providing high-bandwidth and low latency communications over a small area, and global services providing lower bandwidth to a wider area [1]. Currently, no single technology is capable of simultaneously providing high bandwidth to a large number of mobile users over a wide area, and advances in the near future are unlikely to change that given fundamental constraints imposed by power, and by signal-to-noise and signal-to-interference ratios [2].

Heterogeneous wireless networking is a paradigm that can overcome this limitation by using a combination of existing technologies. It is expected that such networks will be deployed extensively in the near future to redefine our understanding and expectations of wireless communications by simultaneously providing universal coverage and high bandwidth access where available [3–5]. Heterogeneous Wireless Networks (HWN) consist of several layers of different overlapping access technologies. This multi-layer architecture provides users with the option of choosing between available services based on traffic profiles, mobility patterns, and Quality of Service (QoS) preferences.

Currently, the most commonly deployed HWN architecture is the integration of 3G at the overlay with IEEE 802.11 at the underlay [6]. An example is shown in Fig. 1.1. At the top layer,
the overlay network provides wide-area service, and at the bottom layer, the underlay network provides local-area service. Overlay network availability is, in general, higher as compared to the underlay network, but it provides lower data-rates at a higher cost [7]. It is anticipated that this new service model will become widely popular, and as such, many new mobile hand-sets are equipped with dual-mode interfaces (e.g., 3G/IEEE 802.11).

A major technical requirement for HWNs, before they can be fully incorporated into future wireless infrastructure, is that they support QoS for multimedia services. QoS provisioning in these networks is challenging due to the diversity in existing technologies and the presence of mobile users with different communication preferences. In addition, to achieve an acceptable QoS, performance challenges have to be addressed at two levels. On one level, we are concerned with per packet performance metrics such as packet drop rate and delivery latency and jitter. On another level, we want to guarantee that ongoing calls will not be dropped when a mobile user moves to another coverage area.

When a mobile user crosses the coverage area of a given base station or WLAN AP, it has to disassociate from its current network, and quickly associate with the base station in the target cell. This is referred to as a handoff event. Handoff between two access technologies operating

---

**Figure 1.1: Sample network architecture.**

---

---

---
at different layers is called vertical handoff (VHO) [3]. The increasing demand for higher data rates will force network designers to use smaller coverage areas and a larger number of base stations. This will make handoff a more frequent event in future networks.

Vertical handoff from a lower to an upper layer is undesirable because resources at higher layers are more competitive [1]. At higher layers, the transmitting station would interfere with a larger number of stations resulting in less spectrum efficiency. As well, accommodating a VHO requires seamless transition of an ongoing call between two technologies with potentially non-matching QoS specifications. All of this adds up to make proper handling of vertical handovers one of most resource-intensive challenges for a network management module. Here, while the goal is to maximize performance, a fine balance must be maintained between over-reservation of resources and a high handoff drop rate due to under-reservation.

1.1 Motivations

Call admission control (CAC) schemes are used in wireless networks to achieve a desired QoS level by providing a framework to reserve and prioritize resources in a resource-sharing system. A CAC algorithm dictates the decision to accept or reject calls or handoff requests. Within the context of homogeneous cellular networks CAC schemes are extensively studied. These schemes can be classified into near-optimal heuristics [8] [9] and decision-theoretic optimal methods [10–12].

Although heuristic methods have advantages such as simplicity and low computational complexity, they have two major drawbacks. First, these approaches, being ad-hoc in nature, provide little insight into the structure of the problem. Thus, they can not be easily generalized and applied to similar problems. Second, in complicated systems such as HWNs, the maximum achievable performance is not known and heuristic algorithms give no clue on how well the current performance compares to an optimum performance. This makes it difficult to know if it can be improved any further, and if so to what extent.
Decision-theoretic approaches, while not impacted by the above issues, do not scale well as the system size grows. In the design of these algorithms, Dynamic Programming (DP) [13] and Markov Decision Processes (MDP) [14] are generally used. According to [15], in Markov decision problems, the size of search space grows exponentially with the system capacity and the number of call classes. In practice, for almost all realistic modelings of HWNs, the computational load of finding an optimal policy by MDP algorithms is prohibitively high.

A more effective use of DP-based methods is to obtain structural results for optimal control problems [15–19]. In structural results, a DP formulation is used to characterize the structure of possible optimal policies. Then, knowledge of policy structure can be exploited to design very efficient numerical methods to find the optimal policy [20].

Our main motivation in this thesis is to derive generic principles as to how HWNs should be controlled and designed efficiently. To this end, we will build a decision-theoretic framework to formulate our problem. The framework will be used to find structural results characterizing optimal control policies that can maximize the utilization of HWNs while maintaining a desired QoS level.

1.2 Contributions

This thesis will study optimal resource planning and dynamic resource management in HWNs. We will propose methods as to how HWNs should be deployed, and present control algorithms that can be used to utilize available resources more efficiently. We will highlight our main contributions as follows:

- We will examine the optimal deployment of HWNs and propose a mobility-aware network planning optimization to minimize the rate of upward vertical handoff events and to maximize the total number of users supported by the network. The optimal placement of Access Points (AP) with respect to these two objectives is formulated as an integer programming problem. Our results will show that considering the mobility pattern in the
planning phase of network deployment can significantly improve infrastructure performance [21].

• We will investigate optimal admission control policies employed to maintain QoS in HWNs. A decision theoretic framework for the problem will be derived using a dynamic programming formulation. This framework will lay the foundation for the main body of work presented in this thesis.

• We will prove that for a two-tier wireless network architecture, consisting of a cellular network overlay and a single WLAN AP, the optimal policy has a two-dimensional threshold structure. Moreover, this structural result will be used to design two computationally efficient algorithms, Structured Value Iteration and Structured Update Value Iteration. These algorithms can be used to determine the optimal policy in terms of thresholds. Although the first algorithm is closer in its operation to the conventional Value Iteration, the second algorithm has a significantly lower complexity [22, 23].

• For the case where underlay is a complex WLAN mesh network, we will develop a Partially Observable Markov-Modulated Poisson Process (PO-MMPP) traffic model to characterize the overflow traffic from the underlaying mesh to the overlay. This model captures the burstiness of the overflow traffic under the imperfect observability of the mesh network states. Then, by modeling the overlay network as a controlled PO-MMPP/M/C/C queueing system and obtaining structured decision theoretic results, it will be shown that the optimal control policy for this class of HWNs can be characterized as monotonic threshold curves. Moreover, these results will be used to design a computationally efficient algorithm to determine the optimal policy in terms of thresholds [24].

• We have developed a flexible discrete event network simulator using object-oriented design concepts. It will be used to conduct extensive simulation experiments to verify the correctness of modeling assumptions, and to gauge the effectiveness of proposed control methods. Results suggest that, in both cases and for all practical parameter sets,
the algorithms converge at the overall optimal policy. Furthermore, numerical results will show that the proposed algorithms are efficient in terms of time-complexity and in achieving the optimal performance by significantly reducing the probability of dropped and blocked calls.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter 2, we will briefly overview previous research studies conducted on QoS provisioning in HWNs, and applications of decision theory and structural results in CAC for wireless networking systems. In Chapter 3, we will review dynamic programming and Markov decision processes, and present the mathematical framework used in the rest of this thesis. That chapter will also recount a number of essential numerical techniques and methods used in later chapters, and will discuss the discrete event simulator that we developed for performance studies. Optimal deployment and planning for HWNs will be addressed in Chapter 4, in which the problem of incremental upgrade of a WLAN deployment will be considered, and an optimization framework will be presented which shows the importance of taking mobility management into account at the networking planning stage. In Chapter 5, we will study the problem of optimal CAC for a HWN consisting of a cellular network overlay and a single WLAN AP. Structural results will be derived and efficient algorithms will be proposed to maintain network QoS. In Chapter 6, we will extend the result of Chapter 5 further by studying the optimal CAC for the case where underlay is a WLAN mesh network. We will characterize the overflow process, and use it along with our DP framework to obtain structured control policies. Numerical results will highlight the efficiency of proposed algorithms and the resultant performance gains. Our concluding remarks will be given in Chapter 7.
Chapter 2

Related Work

In this section, we will briefly review relevant research studies conducted in the past.

2.1 QoS and CAC in Heterogenous Wireless Networks

To the best of our knowledge, many aspects of QoS provisioning in HWNs have yet to be addressed by the research community and there remain many unresolved problems which have to be solved before HWNs can be fully employed in practice. Based on the type of challenge addressed by each approach, we can classify existing studies into:

1. Studies investigating handoff execution delay, aiming to minimizing the time it takes to execute steps involved in a handoff process such as acquiring a network address through DHCP, association/authorization and updating Mobile-IP/SIP records [25–28].

2. Studies addressing the issue of when to initiate a handoff request based on criteria such as signal strength and user mobility pattern [29, 30].

3. Studies addressing the question of which network to choose for new and handoff users when there are many available networks to connect to [31–35].
4. Studies focusing on architecture and protocol design for the integration of different wireless networks [6, 28, 36–40].

5. Studies on reservation and provisioning of communication resources in a destination network, aiming to minimize the chance of a sudden termination of ongoing calls due to handoff rejection [37, 41–43].

A segment of current research on HWNs focuses on designing handoff management protocols and measurements of delays in different steps of performing a handover, with the goal of guaranteeing the successful completion of the handoff procedure within certain time limits [25, 26, 28]. Here, the idea is that if handoff to the next network happens fast enough, the handover experience will be seamless from user’s perspective. In [25], the authors propose a mobility management system consisting of two entities: the connection manager (CM) and virtual connectivity (VC). The role of the CM is to pro-actively make handover decisions according to roaming events and the role of VC is to make the mobility transparent to applications by using a local connection translation. The joint work of CM and VC components will result in seamless handovers. In [27], the authors consider challenges in vertical handoff between 802.11 and 802.16 wireless access networks and propose a method to reduce signaling cost and handoff delay.

Another related research trend addresses challenges in deciding when to initiate a handover and which network to choose [29, 32]. Previous studies on micro/pico cellular networks have been found to be applicable to these scenarios [44]. Several criteria such as mobile user’s speed, cells’ geographical condition and the price a user is willing to pay for communication services play a role in deciding which network to connect to. The authors in [32] propose and compare two assignment strategies. One strategy is based on the user’s velocity and the other is based on the amount of data to be transmitted. An assignment strategy is a policy that determines to which network a user has to be associated. They show that both strategies yield the same minimum average number of users in the system and the same minimum expected
system load. Another interesting network selection scheme is proposed in [35], in which the authors consider the amount of resources that are required to support a connection, and partition the cell coverage based on that figure. This is then used along with a user’s load profile to set proper network preferences yielding optimum performance and utilization.

In [29], the issue of deciding when to initiate a vertical handoff based on criteria such as wireless channel state, network layer characteristics and application requirements is considered. The authors propose an application-based signal strength threshold adaptation to improve performance with regard to signalling load and available capacity. The authors in [30] use an MDP-based scheme to determine the optimal timing to initiate a vertical handoff. They define a link reward function to capture the utilization of communication resources and a signalling cost function to capture the signalling overhead of performing VHOs. Afterwards, they try to achieve a balance between these two by maximizing total reward per connection. In [34], the authors use the theory of evolutionary games to study the dynamics of network selection in HWNs. Competition among users in utilizing limited wireless resources in considered, and two algorithms, one based on centralized control and another one based on reinforcement learning, are proposed for optimal network selection.

Not surprisingly, most of the existing contributions so far are dedicated to architecture and protocol design for HWNs. The heterogeneity of these networks makes the inter-operation and coordination between different layers challenging. One particular complication is that often different and possibly incompatible QoS provisioning schemes are used in various wireless access technologies. For example in IEEE 802.11 WLAN, QoS is achieved by differentiation mechanisms specified in IEEE 802.11e standard, which provide probabilistic QoS guarantees, in contrast to strict TDMA-style QoS offered in GSM Cellular or WiMAX networks. In [40], the drawbacks of traditional handoff protocols and their incompatibilities with emerging HWNs are discussed. For such networks, the authors then study protocol design alternatives that can help in managing vertical handovers seamlessly.
There are only a limited number of studies on QoS provisioning with respect to resource reservation in HWNs. In a few of these studies conventional guard-channel policy is used and the available channels are divided between handover and new calls coming from different layers. In [37], admission control for voice and data services is considered. There, knowledge of wireless overlay structure, non-uniform traffic distribution and user mobility is used to find admission regions. In [41], multimedia traffic flows are divided into real-time and non-real-time categories, and then a link-layer differentiation mechanism is proposed to maintain QoS. A similar approach is used in [42], where the authors consider the integration of 3G and IEEE 802.16e networks, and propose a CAC algorithm differentiating real-time and non-real-time connections. Then, they use a joint packet and connection optimization to maximize QoS. In another work [43], minimization of linear cost functions is used to find the optimal number of guard channels to be used for handoff provisioning.

In all of the aforementioned studies the reservation scheme is chosen a priori, i.e., it is assumed that a fixed threshold policy is capable of achieving the optimal performance. This is in contrast to the approach taken in this thesis in which we do not make any assumption about the structure of the optimal policy, and rather, we try to rigorously characterize it by allowing it to emerge naturally out of our decision theoretic framework.

### 2.2 MDP Optimization of CAC for Wireless Networks

Decision theoretic optimization for Markovian processes is a well-known stochastic control method [45]. The Markov property allows for a significant reduction in tabular programming complexity and in some cases makes it possible to obtain structural results.

Previously, MDP methods have been used in the literature to find optimal CAC schemes for wireless networks. In these studies several optimality criteria are considered. The most common ones are minimization of a total cost (objective) function and minimization of the blocking probability given some hard constraints on dropping probabilities. Several methods
such as Value Iteration (VI), Policy Iteration (PI) and Linear Programming (LP) methods are developed to solve general MDP problems [14].

Most of the existing research uses LP in order to find the optimal CAC policy [11, 12], although VI has also been used in the past [46]. LP has advantages such as formulation simplicity and capability of incorporating probabilistic constraints into optimization over VI and PI. However, it can only be used to solve infinite-horizon average cost problems while PI and VI can be used to solve finite-horizon problems as well. The other advantage of VI/PI over LP is its lower computational cost.

For example, the authors of [12] use LP to obtain the optimal CAC rule which maximizes the service provider’s revenue. The authors consider the case in which each call class can have a different bandwidth requirement. This has applications in multimedia networks in which video, voice and data streams can coexist, having their own QoS and bandwidth restrictions. They also integrate statistical constraints on the maximum tolerable handoff dropping probability into the optimization formulation.

In [11], two modeling approaches - considering the network as a whole or isolating one cell - are used in LP formulation of CAC optimization. The authors provide a solid discussion on the abstraction/isolation level at which such networks have to be modeled. It is explained that handoff arrival to a cell depends on the state of neighboring cells, which if integrated into state variable, results in an exhaustively large state space and an impractically large search space for optimal policies. Afterwards, through simulation studies, the authors claim that the reduced-complexity modeling approach - in which one single isolated cell is considered - provides accurate-enough results.

It is known that in Markov decision problems the number of CAC policies (search space) grows exponentially with the system size. This can hinder the application of optimal CAC schemes in practical scenarios with large state spaces. As a remedy, one common modeling approach, used in all aforementioned studies, is to isolate one cell from the rest of the network to avoid excessive complexity in state space [10, 46].
Despite employing many simplifying assumptions and modeling techniques, the computational cost of using MDP methods directly in finding optimal control policies for call admission is very high. As such others have tried to solve these problems more efficiently by obtaining structural results that will be discussed in the next section.

2.3 Structural Result for CAC

Originally, structural results have been used in operations research studies. As explained previously, obtaining structural results allows the use of efficient numerical algorithms in dealing with complex optimization problems such as those arising in optimal admission control.

One of the earliest works is “the stochastic knapsack problem” in [18]. The conventional knapsack problem is considered under stochastic arrival and object residence times. It is assumed that admitting an object of class $k$ will result in accumulation of revenue at rate $r_k$. Then the authors try to find an optimal admission policy dependent on the current state of the knapsack to maximize the average revenue. They show that the optimal policy is threshold-based when $K$, the total number of call classes, is 2.

In another related work [19] structural results are derived for the Policy Iteration (PI) algorithm. The authors consider the problem of optimal control of a queueing system with heterogeneous servers. Without proof, they assume that the problem has a two threshold optimal policy $\pi = \{M, m\}$ in which $M$ is the upper threshold and $m$ is the lower threshold for slower server turn on/off. A heuristic method is then proposed to find $M$ and $m$.

A similar work has been presented in [16], in which event-based dynamic programming is used to formulate call admission control of multiple classes in a resource-sharing system. They show that customer classes might be ordered in some cases. When classes are ordered, if admitting one class is optimal, then admitting a call of a more rewarding class is optimal as well. One major contribution is that they use a fluid flow model to study CAC for large capacity systems and show that when service time distributions are the same for all classes a
simple trunk reservation policy will be optimal, and modeling using only a one-dimensional state-space will suffice.

In a recent work [15], structured CAC policies for resource-sharing systems are examined, and it is stated that finding the optimal structured policy in the general case is a complex unsolved combinatorial optimization problem. The authors consider two major classes of structured policies, namely, reservation and threshold policies, and propose fast search algorithms to find the policy parameters and provide rigorous proofs for the convergence of the algorithms. Moreover, it is shown that the resultant optimal or near-optimal policies are superior to policies given by commonly used heuristic algorithms.

In addition to these studies, which mostly deal with CAC at an abstract level, the problem of structural results has been studied in the context of cellular wireless networks. A closely related work to this thesis is [10], in which the optimal CAC for a single cellular BS is considered. QoS constraints are expressed in terms of dropping and blocking probabilities and three optimization problems called $\text{MINOBJ}$, $\text{MINBLOCK}$ and $\text{MINC}$ are defined. $\text{MINOBJ}$ refers to the minimization of a linear objective function of the two rejection probabilities. $\text{MINBLOCK}$ is the optimization problem in which the blocking probability is minimized under some hard upper-bound constraints on the dropping probabilities. Also, $\text{MINC}$ refers to the minimization of the number of required channels subject to hard constraints on each rejection probability.

$\text{MINOBJ}$ and $\text{MINBLOCK}$ are the most widely accepted optimality criteria in the study of CAC algorithms. The main advantage of $\text{MINBLOCK}$ lies in the fact that it can guarantee some upper bounds on the dropping probabilities. This can also be achieved by $\text{MINOBJ}$ by adjusting cost ratios. Furthermore, $\text{MINBLOCK}$ has the drawback of not taking into account how much resource is wasted in reservation to achieve those bounds [46].

Moreover, the authors of [10] show that the optimal solution to $\text{MINOBJ}$ for a single cellular Base-Station (BS) is the well-known guard-channel (GC) policy. Then, knowing that the guard-channel policy is fully determined by a single threshold, an efficient method based on bisection search is proposed to find it. Note that a single threshold can be found simply using bisection
search while the original problem involved integer programming. Also, a new policy called Fractional Guard Channel Policy (FGC) is proposed and shown to be the optimal solution to MINBLOCK. The basic difference between GC and FGC is that in FGC a non-integral number of channels are reserved. In other words, in FGC when the system state is at the threshold level, new calls are only blocked according to some given probability, rather than a zero/one decision scheme.

2.4 Overflow Modeling

In Chapter 6, we consider the integration of a wireless mesh network with cellular overlay and model the handoff arrival to the overlay as an overflow process. In this section, we present previous research efforts on the modeling of overflow streams.

The earliest studies on overflow modeling date back to research on the performance of circuit-switched PSTNs where arrival to busy trunks would overflow to secondary circuits. Many studies were devoted to this issue and several fundamental contributions were published [47–49]. Later, in the context of alternate routing and hierarchical cellular networks, the issue of overflow modeling was revisited again [50]. While complete and exhaustive modeling of overflow stream is possible (and has already been done [51, 52]) for a number of systems, it has a very high computational cost. As such approximate models have been used to find performance measures of interest. We have identified four major approaches:

- Modeling overflow as a Poisson process [50]
- Using Equivalent Random Method and its enhancements [47–49]
- Using Hayward’s approximation and its enhancements [53–55]
- Using interrupted and Markov-modulated Poisson processes [56–68]

Initially, aggregate overflow traffic in hierarchical wireless networks was modeled as a Poisson process. However, it was observed that this approximation was inaccurate, and by failing
to capture the burstiness of overflow traffic, it underestimates the blocking rate of overflow calls [54]. Several methods were proposed to tackle this issue. Equivalent Random Method (ERM) [47], one of the earliest contributions, provides a framework to find the “equivalent random” load \( \hat{A} \) and server group size \( \hat{S} \) that results in a given mean and variance for the overflow process. Here, mean and variance refers to the mean and the variance of the number of calls in a virtual overlay group with infinite number of servers.

To illustrate how ERM can be used, consider a scenario in which multiple streams of traffic overflow from a number of underlay queues to a shared server group. If we know the mean and the variance of the aggregate overflow traffic, we can use ERM to find \( \hat{A} \) and \( \hat{S} \) such that if we replace the entire underlay network with a single server group \((M/M/\hat{S}/\hat{S})\) of size \( \hat{S} \) and an incoming load of \( \hat{A} \) Erlangs, then the mean and the variance of overflow to the shared group will stay the same. This equivalent model, which is a reduced and simplified version of the original network, can be analyzed at a lower computational cost.

Another notable method is Hayward’s approximation [53]. For loss systems and under Markovian assumptions, we can use Erlang-B formula to find the blocking probability of a server group. When the incoming load is not Poisson, such as when it is a bursty overflow process, this formula does not hold. Hayward’s approximation provides an adjustment to the incoming load by taking its peakedness into account [55]. A number of enhancements are available. In [55], the authors extend Hayward’s method to handle multiservice overflow traffic. Multiservice traffic refers to having several call classes with potentially different arrival and service rates. An efficient method named Multiservice Overflow Approximation (MOA) is proposed, which uses the matching of blocking probabilities to find mean and variance of overflow traffic, and utilizes a modified Hayward’s approximation to find loss probabilities.

Hayward’s original method is only applicable to loss networks which have no queuing capacity, i.e., a call will be dropped if it cannot be accommodated. In [54], Hayward’s method is extended further to cover server groups with waiting rooms. They find the Laplace transform of the interoverflow time distribution, and use it to study the peakedness of the overflow process.
This is used to propose an approximation for the blocking seen by overflow traffic. It is also shown that increasing the queue size can reduce the peakedness of overflow stream.

Both ERM and Hayward’s method provide a framework to take the first two moments of an overflow process into account in finding blocking probabilities. Another enhancement to overflow modeling was introduced by using Interrupted and Markov-modulated Poisson processes (IPP and MMPP). IPP can be used to match the first three moments. While IPP is a limited case of MMPP, its simplicity, tractability and accuracy have made it very popular [56]. Several studies, which compare the results of IPP and MMPP (an exact analysis under Markovian assumption) modeling, confirm that IPP yields accurate-enough performance metrics compared to the computationally expensive but otherwise exact MMPP approach [69]. It is also confirmed that IPP results outperform those of ERM and Hayward’s approximation.

An IPP(λ, ω, α) has a two-state underlying Markov chain as depicted in Figure 2.1 and is identified by three parameters: λ instantaneous traffic rate when the underlying markov chain is ON, ω rate of transition from OFF to ON, and α rate of transition from ON to OFF.

IPP modeling of overflow traffic first appeared in [56]. The authors give an iterative expression for the factorial moments of the number of busy servers in a server group of infinite capacity (M/M/∞/∞) with an overflow arrival process characterized by an IPP. Then, the first three moments are matched with the factorial moments derived from an exact analysis of the number of busy servers with arrival coming from a blocking server group. This fully identifies the equivalent IPP by yielding λ, ω and α. It is also discusses the ways in which ERM can be used, if we know the mean and the variance of the overflow traffic, to find these parameters.

Figure 2.1: Underlying Markov chain for IPP.
Numerical experiments confirm that using IPP produces very accurate results. Additionally, they provide a closed form expression for interoverflow time distribution, which is a mix of two exponentials.

MMPP models have been used in the literature to study the performance of hierarchical networks and alternative routing schemes [67–69]. In [62], overflow traffic from a set of primary queues to a secondary queue is characterized by using an MMPP. The author presents a novel framework to find the performance of the secondary queue by modeling it as a MMPP/M/C/C+K system. When multiple independent overflow streams are joined, a Kronecker sum can be used to find the transition rate matrix (Q) of the aggregate stream. It is shown that this matrix, despite having a large number of states, is band-limited, and block Gauss-Seidel iteration is used to efficiently find its stationary distribution. The framework is further used to obtain several performance measures of interest including steady-state queue length distribution and blocking probability of arrivals to the secondary queue.

A common practice in studies involving multiple primary queues is to model each as an IPP, and then to use an MMPP to represent aggregate overflow [58, 60, 66, 70]. For this modeling to be tractable, it is almost always assumed that IPPs are statistically independent. While this probably holds true for a number of networking systems, it is not generally correct. It is, however, argued that in many scenarios using MMPP to model aggregate overflow, when underlying IPPs are not totally independent, yields acceptable approximates. This is investigated in Section 6.9. We refer interested readers to the results given in [61] and in Section IV of [62].
Chapter 3

Backgrounds

In this section, we will present the mathematical framework used in the next chapters, and will elaborate on the discrete event simulation engine developed to study the performance of proposed CAC schemes.

3.1 Dynamic Programming Framework

In this section, we will present the mathematical framework used to formulate our dynamic programming problem, and we explain some essential numerical techniques that has to be used along with MDP such as the method of fictitious decision epochs and uniformization.

Dynamic programming optimization using Markov Decision Processes (MDPs) is a well-known discrete-time stochastic control method [45]. The Markov property allows for significant reduction in tabular programming complexity and in some cases makes it possible to obtain structural results. An MDP is determined by four components: state space $S$, action space $A$, state transition probabilities $P$ (or transition rates $Q$ in case of continuous-time processes), and a cost function $C$. The solution to an MDP is called a policy or rule. A policy maps the state space to actions $\Psi : S \rightarrow A$, such that a given optimization goal is achieved. A large class of policies, in which the decision is independent of time given the system state, is called stationary policy.
In every state there are a number of possible actions. Once an action is taken it will incur some cost that will be added to the total system running cost. In our event-based DP, we associate costs to undesirable control decision events. These costs could, for example, correspond to the dropping or blocking of incoming calls. The selected action also dictates a set of state transition probabilities out of the current state.

The MDP performance criteria (system cost function) can be formulated with respect to finite or infinite horizons, and for average-cost or discounted-cost problems. In this thesis we are generally interested in minimizing the average cost per unit time for an infinite-horizon non-discounted problem. This reflects our concern about long-run QoS performance. Additionally, because the decision epochs can be at any randomly distributed time, a Semi-Markov Decision Process (SMDP) model is used [71].

Depending on the problem size, the computation cost of finding an optimal policy can be relatively high making it essential to use efficient algorithms. There are two general approaches to solving an MDP problem. The first approach uses Dynamic Programming. There are a few variations of this type such as Value Iteration (VI) [72], Policy Iteration (PI) [73], Modified Policy Iteration [14] and Prioritized Sweeping [74]. The second approach involves a Linear Programming formulation, but it is only applicable to cases with finite action and state spaces. While not the most efficient, it is also possible to solve an MDP by using combinatorial search given an explicit formulation of objective function (cost criterion).

The Value Iteration algorithm is the most frequently used technique to solve MDP problems. It is based on the Bellman-Ford iterative equation [14],

\[ V_n(s) = \min_{a \in A(s)} \{ c_s(a) + \sum_{t \in S} P_{st}(a) V_{n-1}(t) \}. \]  

Note that this equation is backward in time, such that \( V_0(s) \) is the cost at the last time step. In every iteration \( V_n(s) \) is calculated for \( \forall s \in S \). Here, \( S \) is the state space, and \( A(s) \) is the set of possible actions at state \( s \). \( P_{st}(a) \) is the transition probability of going from \( s \) to \( t \) having taken action \( a \), and \( c_s(a) \) is the cost of taking action \( a \) in state \( s \). A more formal presentation of VI
for average cost problems (non-discounted case) is given in Algorithm 1. Later in Chapters 5 and 6, we will use VI as a foundation, and propose several improvements to enhance its speed and memory requirements when used to find CAC algorithms.

**Algorithm 1** Value Iteration Algorithm.

1: Initialize $\forall s \in S: V_0(s) = 0$

2: $n := 1$

3: Update $V_n(s)$ and $\pi_n(s)$ for $s \in S$:

4: $V_n(s) = \min_{a \in A(s)} \{c_s(a) + \sum_{t \in S} P_{st}(a) V_{n-1}(t)\}$

5: $\pi_n(s) = \text{argmin}_{a \in A(s)} \{c_s(a) + \sum_{t \in S} P_{st}(a) V_{n-1}(t)\}$

6: Find:

7: $M_n = \max_{t \in S} \{V_{n+1}(t) - V_n(t)\}$

8: $m_n = \min_{t \in S} \{V_{n+1}(t) - V_n(t)\}$

9: Set $d := \frac{M_n - m_n}{m_n}$

10: if $d \leq \epsilon$ then

11: Return Policy $\pi_n$

12: else

13: $n = n + 1$

14: Go to step 3

15: end if

Here $\epsilon$ is a pre-specified tolerance level for convergence error. It is well known that for average-cost problems with finite $S$ and $A$, and time-invariant transition probabilities, the optimal policy is stationary. Furthermore, we are only interested in stationary policies that result in irreducible chains. For such irreducible and aperiodic MDPs, it is shown that VI converges in a finite number of iterations (Theorem (6.6.2) [71]), and that the upper and lower bounds of $V_{k+1}(s) - V_k(s)$ converge to the optimal average cost per unit time when $k \to \infty$. More
formally, given
\[
M_k = \max_{t \in S} \{ V_{k+1}(t) - V_k(t) \}
\]
\[
m_k = \min_{t \in S} \{ V_{k+1}(t) - V_k(t) \},
\]
the optimal cost \( g_\Psi \) is equal to \( g_\Psi = \lim_{k \to \infty} M_k = \lim_{k \to \infty} m_k \). Structural results for \( V_k(s) \) as defined in (3.3) will hold for the optimal per-unit-time average cost function if the underlying Markov decision process is irreducible and aperiodic (Theorem (6.6.1) [71]).

It can be seen from equation (3.1) that the computational complexity of evaluating the cost function in every step depends on the density of \( P_{st}(a) \) matrix. If for a given \( s \), the number of possible next states \( t \) is large, then the summation will have a large number of non-zero terms. One intelligent technique to make VI faster is to use the method fictitious decision epochs [71] in formulating state transition probabilities \( P_{st}(a) \). When times between decision epochs are exponentially distributed we can reduce the computation cost by introducing fictitious decision epochs at which no real decision has to be made. These correspond to departure events when no action is taken. Through this technique, at every decision epoch either real or fictitious, the system state can only change to adjacent states, making many terms in \( P_{st}(a) \) zero. However, to keep track of the epoch type we have to extend the state space by one dimension. The increased computation cost due to this enlarged state-space is compensated by the reduction in \( P_{st}(a) \) density.

As an example in using fictitious decision epochs, consider a queue serving \( i \) users. If we were to look at the system state at the next arrival epoch, we may find it having 0 \ldots i users, depending on how many users have been served in that duration, and if the last arrival was admitted or rejected. Now, let us redefine the the state as \( s = (i, k) \), where \( k \) identifies the type of last decision epoch (1 for arrivals, and 0 for departures). If the current system state is \( s = (i, k) \), then at the next event, it can only assume one of these values \( s' = (i, 0) \), \( s' = (i - 1, 0) \), \( s' = (i, 1) \) and \( s' = (i - 1, 1) \). The result is that in the matrix representing \( P_{st}(a) \), most term will be zero except for when \( t = s' \). It it clear that if the system maximum capacity (i.e., the highest value \( i \) can assume) is large enough, the computational and storage
savings could be significant.

In the remainder of this section we present a general framework that will be used throughout the rest of this thesis. Let us denote $V_k(s)$ to be the minimum expected cost function for a $k$-stage problem with the initial state $s$. And let $E_A$ denote the set of arrival events that are controllable and can potentially be rejected at some costs, and let $E_N$ denote the set of internal transitions and departures for which no action needs to be taken. Using the uniformization technique [71], we can write $V_{k+1}(s)$ recursively as

$$V_{k+1}(s) = \frac{1}{v_{\text{max}}} \left\{ \sum_{e \in E_A} q_e \left[ \min\{ \Delta V_k(\psi_e s), C_R(e) \} + V_k(s) \right] + \sum_{e \in E_N} q_e V_k(\psi_e s) + \left( v_{\text{max}} - v_{\text{out}}(s) \right) V_k(s) \right\}$$

(3.3)

where $q_e$ is the rate of transition for events of type $e$, $C_R(e)$ is the cost of blocking event $e$, $v_{\text{out}}(s)$ is the rate of going out of state $s$, and $v_{\text{max}}$ is the uniformization parameter such that $v_{\text{max}} \geq v_{\text{out}}(s)$ for every $s$. Here, the operator $\psi_e$ acts on state $s$ and returns the resultant state if event $e$ was to be admitted. The Δ symbol for an operator $\psi$ is defined as

$$\Delta V_k(\psi_e s) = V_k(\psi_e s) - V_k(s).$$

(3.4)

Equation (3.3) consists of three terms, each reflecting one possible event. The first term accounts for arrivals to the cluster, and the second term accounts for departures and internal non-controllable transitions. The last term is due to the uniformization technique where staying in the same state is possible. In the first term, $\Delta V_k(\psi_e s)$ is the cost of admitting a call whose arrival is triggered by event $e$. If this cost is less than the blocking cost $C_R(e)$, then the call will be admitted; otherwise it is rejected. More general information on similar frameworks can be found in [10, 75].

### 3.2 Discrete Event Simulation

In this thesis we will adopt a novel approach to study the QoS performance and computational efficiency of different admission control policies. Our simulation methodology has two stages.
In the first stage, an efficient code developed in MATLAB is used to find an optimal control policy. Once we have the policy, we need to know how it would perform if it were to be used in a wireless network setting. In the second stage, this policy is fed to a discrete event simulator developed in C++, named Wireless Network Simulator (WNS). This two-phase method is depicted in Figure 3.1. Discrete event simulation is an efficient and popular approach to simulate networking systems at a micro-level involving calls and packets.

WNS, by simulating a simplified version of real networking components, allows for using actual models such as wireless base stations, WLAN access points and mobile users moving from one coverage area to another. It works by maintaining a sorted list of time points at which we expect a consequential event such as a call arrival, departure or handover. Then, starting from a reference point in time, it goes from one event to the next while updating the system status proportionately.

The most common practice in implementing software applications of this nature is to use an object-oriented design. Figure 3.2 shows the class hierarchy we have used. Here, all objects are derived from a generic simulation objects named SObject, which is capable of triggering and receiving events. From SObject, we derive three main types:

- **S.Wireless_Call_TS**: generic traffic source
- **SWBS**: generic base station
- **SWCall_Sink**: generic traffic sink

In this context, traffic refers to ongoing calls. In a normal cycle of events, a call is generated in a traffic source and is sent to a base station. This corresponds to when a mobile user initiates
a call while residing in that station’s coverage area. After some time, the call either moves to another station, or terminates. A terminated, blocked or dropped call is sent to a traffic sink. This helps to collect statistics, and to release allocated resources.

From SWBS, we derive two types of SW_CAC_BS and SW_TResv_BS. Here, SW_CAC_BS represents a wireless station with exponential mobility model (calls move from this station to a neighboring station within an exponentially distributed time) and an admission control decision unit. SW_TResv_BS represents a base station exercising trunk reservation (guard channel) policy. In contrast to SW_CAC_BS which can employ complex CAC policies, SW_TResv_BS uses a simple threshold-level scheme to make CAC decisions. From SW_CAC_BS, we also derive a mesh-network capable variant called SWM_CAC_BS.

Figure 3.3 presents what is referred to in the Unified Modeling Language (UML) as a collaboration diagram depicting relationships and interactions among different components for SWM_CAC_BS. In this figure solid blue arrows indicate inheritance relations, and purple dashed arrows visualize membership or usage relations.

Two other notable components in figure 3.3 are the Simulator and Event_Scheduler. The Simulator provides the main framework connecting different networking components (piping), keeps track of time and events and controls the simulation flow. The Event_Scheduler maintains a sorted list of events in a priority-queue and can be called by Simulator to start executing events one after another.
Figure 3.3: Collaboration diagram for SWM_CAC BS.
Chapter 4

Optimal Deployment of HWNs

4.1 Overview

A natural first step in enabling widespread adoption of Heterogeneous Wireless Networks (HWN) is to shed light into how these networks should be implemented. Optimal deployment of wireless infrastructure has been extensively studied in the literature [76–78]. The conventional constraints in the planning of wireless networks are capacity, signal strength and frequency channels. However, efficient deployment of next-generation HWNs requires taking a new important factor, vertical handovers, into account. To the best of our knowledge, this has not been addressed in the earlier studies. In what follows, we will elaborate on why vertical handovers have to be considered in the network planning phase and will present a framework to integrate this constraint into our network optimization1.

Since the radio spectrum is limited, pico/micro-cellular and WLAN architectures will be used more and more in the future to satisfy users’ demand for higher data rates. The result is that as the radius of coverage area shrinks, the crossing of coverage borders, handover, is becoming a major performance criterion. Handoff between two different access technologies is called vertical handoff (VHO) [1]. Vertical handoff from a lower layer to an upper layer is

---

1This chapter is based on our publication in [21].
an undesirable incident, because resources at higher levels are more limited [1].

As mentioned in Chapter 2, provisioning handover calls is a resource-intensive challenge that requires allocation of bandwidth in the destination networks, which can be potentially wasteful. To illustrate why this can impact network planning, consider a scenario in which it might be better to bundle some APs together instead of spreading them over the deployment area. That way, although the APs might not be accommodating the highest amount of traffic, the continuous deployment mitigates the need for frequent vertical handovers. This in turn reduces the amount of resources that had to be reserved at the overlay to accommodate upward vertical handovers. The aggregate effect is that while we provided less capacity in the underlay, it resulted in better utilization of resources at the overlay where resources are more scarce.

Therefore, minimization of vertical handoff rate has to be considered in the network planning. However, this single objective, while necessary, is not enough in defining our optimization problem as there is a clear trade-off between lowering handoff rates and supporting higher amounts of incoming traffic. To address this, here, we will use a cost function that depends not only on the rate of vertical handovers, but also, on the total supported traffic. To demonstrate why this second objective is needed consider that if we were to only use the first one, the algorithm might choose to place APs in the areas with the least amount of traffic, in the worst case, as it would result in lower vertical handoff rates.

In this chapter, we show that for a linear cost function the resultant optimization problem can be solved efficiently by integer programming. This cost function aggregates the cost of supporting vertical handovers and the cost of not accommodating the maximum amount of traffic. The linear cost combination reflects our perception of the underlying costs in the real world and from a service provider’s perspective. Vertical handovers are costly as resources have to be reserved (and potentially wasted) in the overlay to guarantee their uninterrupted transition, and supporting less incoming traffic leads to loss of revenue.

To the best of our knowledge, this is the first work in which the inefficiency of conventional planning methods due to the more frequent handoff events is considered and an integer pro-
Figure 4.1: Sample network structure; APs deployment has resulted in having two independent clusters.

programming framework to minimize the vertical handoff rate is proposed. In what follows, we first present our system model and assumptions and then the integer programming formulation. Finally, performance results and conclusion are presented.

4.2 Background

In [76], the problem of finding an efficient WLAN deployment strategy to maximize the coverage area and to improve the overall signal quality is explored. Generally, these two objectives have different, and often conflicting, solutions. To strike a balance between these two goals a combined objective function approach is proposed in [79]. We adopt this technique in formulating our optimization problem. It is also tried to tackle the problem of incremental AP deployment. The authors in [77] present a series of Internet Transit Access Point (ITAP) placement algorithms to build efficient multi-hop wireless neighborhood networks. The goal is to find the optimal deployment of ITAPs that minimizes the total number of required ITAPs under certain user bandwidth requirement constraints and various wireless link models. In [78], the
CHAPTER 4. OPTIMAL DEPLOYMENT OF HWNs

The problem of optimal placement of wireless relays is investigated, and the authors use a novel numerical method to solve the resultant integer programming optimization.

4.3 Assumptions and System Model

Generally, in the network planning or upgrade process, the goal is to add $N$ new APs to the the underlay architecture which may already have $N_0$ APs. The deployment area is divided into an H-by-W grid space and each block is called a cell. We assume that each cell may contain an AP. The set of adjacent cells with an AP inside them form a cluster. When the planning is done we might have several independent clusters as depicted in Fig. 4.1. Vertical handoff occurs when a mobile user having an active call leaves a cluster and starts to use the overlay networking services. In this chapter our objective is to jointly minimize the total rate of the upward vertical handoff events and to maximize the total supported number of network users.

To reduce the optimal planning complexity and to make the problem tractable a discrete search space is assumed in this thesis. Discrete search space for AP placement is used in many papers [76, 79]. The error introduced by this approximation is negligible considering the uncertainties in traffic characteristics modeling, i.e., the call arrival rate in each area, the call duration distribution and handoff and coverage-crossing probabilities. This error is also overshadowed by the fact that in practice the placement of an AP is highly dependent on the site-specific constraints.

One key observation here is that the user mobility pattern is independent of the infrastructure presence and its variations. Generally, this is true because users are commonly not concerned with how the networking services are achieved and they make their movements decisions independent of that. This allows us to look at the steady-state occupancy distribution of each cell independently of how the final deployment for the underlay network is done. Furthermore, we assume that all arrival and departure processes are memoryless and as such are Poisson processes. As shown in Fig. 4.2, we denote by $\lambda(i, j)$ the call arrival rate to cell
Figure 4.2: Traffic arrival and departure from AP(i,j).

\((i, j)\), the call termination rate by \(\mu(i, j)\) and call handoff rates from cell \((i, j)\) to each of its neighboring cells by \(H(i, j, k)\).

### 4.4 Mobility and Equilibrium state

We divide the vertical handoff minimization problem into two parts. Here, we first use the mobility pattern to find the expected number of users in each possible AP coverage area, and in the next section we use this knowledge to formulate the problem as an integer program. Clearly, the handoff rate out of each cell is proportional to the number of users in the cell.

To find the expected number of users in each cell \(n(i, j)\), we develop a queuing model by considering every cell as an \(M/M/\infty\) queue. Here, it is assumed that the capacity of each cell is unlimited which is not unrealistic knowing that in practice the utilization of underlay APs are normally far below saturation.

In the \(M/M/\infty\) queuing model each call goes to the server as soon as it arrives and no queue is formed and the total waiting time is equal to the system service time. The service time for each call can be computed by considering that a call leaves a cell when it either terminates or handovers to another cell. Let us denote the service time for a call in cell \((i,j)\) by \(S_i(i, j)\) we
can write that

\[
S_t(i,j) = \min(T_t(i,j), T_{h(i,j,1)}, T_{h(i,j,2)}, T_{h(i,j,3)}, T_{h(i,j,4)})
\]  

(4.1)

where \(T_t(i,j)\) denotes the call duration in cell \((i,j)\) before its final termination and \(T_{h(i,j,k)}\) represents the time to handoff to \(k^{th}\) neighbor in the grid. Assuming that all the processes are memoryless, the random variable \(S_t(i,j)\) is exponentially distributed with parameter \(\mu(i,j) + \sum_{k=1}^{4} H(i,j,k)\). Note that in a grid each cell is adjacent to 4 other cells. In Fig. 4.3 a cell and its neighborhood and our numbering convention are shown.

Little’s theorem can be used to compute \(n(i,j)\) given the total arrival rate to cell \((i,j)\) and the average service time as

\[
n(i,j) = \lambda_{\text{total}(i,j)} \times \left(\mu(i,j) + \sum_{k=1}^{4} H(i,j,k)\right)^{-1},
\]  

(4.2)

where the total arrival rate to cell \((i,j)\), \(\lambda_{\text{total}(i,j)}\), is the sum of the call arrivals (new call generation) \(\lambda(i,j)\) and handoff arrivals from adjacent cells. From the traffic equations we have

\[
\lambda_{\text{total}(i,j)} = \lambda(i,j) + \sum_{(i',j') \in S_{\text{nb}(i,j)}} H(i',j',k')n(i',j'),
\]  

(4.3)

where \(S_{\text{nb}(i,j)}\) is the set of cells adjacent to cell \((i,j)\) and \(k'\) is chosen such that cell \((i',j')\) is attached to cell \((i,j)\) by its \(k'th\) side. Equation (4.2) can be used to eliminate \(\lambda_{\text{total}(i,j)}\) as

\[
n(i,j) \times \left\{\mu(i,j) + \sum_{k=1}^{4} H(i,j,k)\right\} - \sum_{(i',j') \in S_{\text{nb}(i,j)}} H(i',j',k')n(i',j') = \lambda(i,j).
\]  

(4.4)

This forms a set of \(H \times W\) independent linear equations which can be solved to find \(n(i,j)\).
4.5 Optimization Problem Formulation

In this section, we present the steps taken to formulate the optimization problem as an integer programming problem. For each cell \((i, j)\), we have a variable \(x(i, j)\) that indicates whether there is an AP in the cell:

\[
x(i, j) = \begin{cases} 
1 & (i, j) \text{ has an AP} \\
0 & (i, j) \text{ does not have an AP.}
\end{cases}
\]  

(4.5)

Then, the total number of users supported by a certain AP deployment can be written as

\[
n_{\text{total}} = \sum_{i=1}^{H} \sum_{j=1}^{W} n(i, j) x(i, j).
\]  

(4.6)

To formulate the upward vertical handoff rate, \(V(i, j, k)\), from cell \((i, j)\) through its kth side, we introduce a variable \(\hat{x}(i, j, k) = x(i', j')\), where cell \((i', j')\) is located on cell \((i, j)\)’s kth side. It is easy to verify that \(x(i, j) [1 - \hat{x}(i, j, k)] = 1\) if and only if \(x(i, j) = 1\) and \(\hat{x}(i, j, k) = 0\), which means \((i, j)\) is a cell on the border of a cluster and handoffs calls going outward through the kth side are vertical upward handoffs. Then we can use \(x(i, j) [1 - \hat{x}(i, j, k)] = 1\) as an indicator of vertical handoffs. For each cell we have

\[
V(i, j, k) = n(i, j) H(i, j, k)x(i, j) [1 - \hat{x}(i, j, k)].
\]  

(4.7)
Furthermore, the total upward vertical handoff rate of a system is

\[ V_{\text{total}} = \sum_{i}^{H} \sum_{j}^{W} \sum_{k=1}^{4} V(i, j, k). \]  

(4.8)

Our goal is to find the optimal AP placement which minimizes \( V_{\text{total}} \) and maximizes \( N_{\text{total}} \) at the same time. However, the two objectives do not necessarily have the same optimal location set. We adopt the approach used in [79] and consider the following objective function:

\[ F_{0} = V_{\text{total}} - \psi n_{\text{total}} = \sum_{i=1}^{H} \sum_{j=1}^{W} n(i, j) \left[ \sum_{k=1}^{4} H(i, j, k) - \psi \right] x(i, j) \]

\[ -\psi \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{4} n(i, j) H(i, j, k) x(i, j) \hat{x}(i, j, k) \]  

(4.9)

Here, \( \psi \in [0, +\infty) \) is the weighting factor. Then the optimization problem is formulated as

\[ \text{minimize} \ F_{0} \]

subject to \[ \sum_{i=1}^{H} \sum_{j=1}^{W} x_{0}(i, j) x(i, j) = N_{0} \]  

\[ \sum_{i=1}^{H} \sum_{j=1}^{W} x(i, j) = N_{0} + N \]  

(4.10)

where \( x_{0}(i, j) \) indicates the existing infrastructure, i.e., if cell \((i, j)\) already has an AP in it, \( x_{0}(i, j) = 1 \); otherwise \( x_{0}(i, j) = 0 \). The first constraint maintains that for every \( x_{0}(i, j) = 1 \), the new deployment must satisfy \( x(i, j) = 1 \), in order to keep the existing APs in the system. The second constraint formulates the constraint that the new system should have \( N \) new APs and \( N_{0} + N \) APs in total.

Notice that \( F_{0} \) is a quadratic function. In order to have a linear objective function, we replace the quadratic terms in \( F_{0} \) with a new variable \( y(i, j, k) \):

\[ F = \sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{4} n(i, j) [H(i, j, k) - \psi] x(i, j) \]

\[ -\sum_{i=1}^{H} \sum_{j=1}^{W} \sum_{k=1}^{4} n(i, j) H(i, j, k) y(i, j, k) \]  

(4.11)
and the optimization problem can be reformulated as

\[
\begin{align*}
\text{minimize} & \quad F \\
\text{subject to} & \quad \sum_{i=1}^{H} \sum_{j=1}^{W} x_0(i, j)x(i, j) = N_0 \\
& \quad \sum_{i=1}^{H} \sum_{j=1}^{W} x(i, j) = N_0 + N \\
& \quad y(i, j, k) \leq x(i, j) \\
& \quad y(i, j, k) \leq \hat{x}(i, j, k) \\
& \quad y(i, j, k) \geq x(i, j) + \hat{x}(i, j, k) - 1
\end{align*}
\]

(4.12)

It is easy to see that the above constraints \( y(i, j, k) = x(i, j)\hat{x}(i, j, k) \), since \( x(i, j), \hat{x}(i, j, k) \in \{0, 1\} \).

Note that there is no coverage continuity constraint in the above optimization problem formulation. However, as it will be shown in the next section, when suitable \( \psi \) is used, continuity is achieved automatically.

### 4.6 Simulation Results

In this section, we study the performance of our algorithm for different preexisting infrastructure installations and different choices of parameters.

#### 4.6.1 Simulation Setup

We generate a 10-by-10 grid as shown in Fig. 4.4. A street is modelled as a vertical path with a relatively high arrival rate passing through in the middle of the region. There is a hot spot on each side of the “street”, representing buildings with a large number of users in them. After solving a set of \( 10 \times 10 \) independent linear equations as described in Section II, we find the average number of users \( n(i, j) \), shown as grey scale in Fig. 4.4.

For the simulation experiments four initial AP settings are considered: 1) no AP is installed in the region; 2) only one AP is installed at the “hottest” spot on the right half of the region; 3) three APs are installed as a cluster around the “hottest” spot; and 4) two APs are installed
around the “hottest” spot and one AP is installed in the hot spot on the left half of the region. The goal is to upgrade the deployment so that the total number of APs would be 15.

### 4.6.2 Effect of Initial Deployment

Figure 4.5 shows the optimization result for the experiment test case 4 where we initially have two clusters and we want to add 12 new APs to the system for $\psi = 40$. As shown in this figure, the optimal solution places two clusters of APs around the two hot spots. Figures 4.5 and 4.6 jointly illustrate how the initial condition impacts the optimization outcome for test cases 3 and 4. As it can be seem from the figures, the initial setup biases the outcome toward placing more APs around already covered regions. This is to be expected as such positioning will decrease the total rate of upward vertical handovers for the system.
Chapter 4. Optimal Deployment of HWNs

Figure 4.5: Upgraded AP deployment: Test 1.

Figure 4.6: Upgraded AP deployment: Test 2.
4.6.3 Effect of Parameters Selection

The value of $\psi$ in the objective function $F_0 = V_{total} - \psi n_{total}$ determines the desired balance between provisioning of handoff events and accommodating more users. When $\psi$ is large, $n_{total}$ is more important and the optimization process will end up choosing cells with larger $n(i, j)$'s. When $\psi$ is close to 0, we are more concerned about $V_{total}$ and this results in choosing cells on the border of the deployment area.

Fig. 4.7 shows the optimal value of $F$ achieved for the four initial settings with $\psi$ varying over a wide range. It can be seen that the optimal $F$ in all cases is lower bounded by the optimal $F$ achieved with no preexisting AP. This is true because the feasible set in the case of no initial AP is the largest. Comparing the results for Case 2 with Cases 3 and 4 shows that the optimum value when there is only one cluster is always better than the case where there are two clusters. This is because unless the new APs connect the two clusters together, the total boundary is always larger than the case when there is only one cluster, and a larger boundary will result in more vertical handoffs. Figures 4.8 and 4.9 show the effect of varying $\psi$ on the components of $F$, namely, the total rate of upward handover, and the total number of accommodated users.
Chapter 4. Optimal Deployment of HWNs

Figure 4.8: Number of total supported users versus $\phi$

Figure 4.9: Vertical Handoff rate versus $\Phi$
4.7 Chapter Summary

In this work, we proposed a mobility-aware network planning optimization algorithm for HWNs. The problem of optimal AP placement is formulated as an integer programming problem. The optimization objective is to minimize the rate of upward vertical handoff events and to maximize the total number of users supported by the network. Our performance results indicate that the optimization algorithm makes efficient decisions regarding AP placement. It is shown that considering mobility pattern in the planning phase of network deployment can significantly improve infrastructure performance. Also, the proposed algorithm can be used to provide design guidelines on incremental upgrade of existing wireless networks.
4.8 Nomenclature
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda(i, j)$</td>
<td>New call arrival rate to cell $(i, j)$</td>
</tr>
<tr>
<td>$\mu(i, j)$</td>
<td>Service rate in cell $(i, j)$</td>
</tr>
<tr>
<td>$H(i, j, k)$</td>
<td>Handoff rate from cell $(i, j)$ to its $k^{th}$ neighbor</td>
</tr>
<tr>
<td>$S_t(i, j)$</td>
<td>Service time for a call in cell $(i, j)$</td>
</tr>
<tr>
<td>$\lambda_{total}(i, j)$</td>
<td>Sum of new call and handoff arrival rates to cell $(i, j)$</td>
</tr>
<tr>
<td>$S_{nb}(i, j)$</td>
<td>Set of cells adjacent to cell $(i, j)$</td>
</tr>
<tr>
<td>$n(i, j)$</td>
<td>Expected number of active calls in cell $(i, j)$</td>
</tr>
<tr>
<td>$n_{total}$</td>
<td>Expected total number calls in the system</td>
</tr>
<tr>
<td>$V(i, j, k)$</td>
<td>Rate of upward vertical handoffs from cell $(i, j)$ though its $k^{th}$ side</td>
</tr>
<tr>
<td>$V_{total}$</td>
<td>Expected total rate of upward vertical handovers</td>
</tr>
<tr>
<td>$F$</td>
<td>Optimization objective function</td>
</tr>
<tr>
<td>$\psi$</td>
<td>Weighting factor to balance optimization goals</td>
</tr>
<tr>
<td>$H \times W$</td>
<td>Dimensions of the discretized space grid</td>
</tr>
</tbody>
</table>

Table 4.1: Symbols used and their description.
Chapter 5

Overlay and IEEE 802.11 Integration

5.1 Overview

In this chapter, we present our contribution on optimal Call Admission Control for a Heterogeneous Wireless Network (HWN) consisting of a cellular network overlay and a single WLAN Access Point (AP). In such HWNs, if a mobile user moves out of the AP coverage area, it will require an upward vertical handover. However, there is no guarantee that this handoff can be accommodated by the overlay network. Efficient allocation of resources to address this challenge is the main focus of this chapter. We base our work on theories in optimal control where dynamic programming methods are used to find the optimal policy to control a random process over time to achieve a certain optimization goal.

As discussed in Chapters 1 and 2, for almost all realistic modelings of networking systems, the computational cost of finding an optimal policy by MDP algorithms is very high and the size of state space grows exponentially with system capacity. Additionally, numerical methods to solve MDP problems are iterative and as reported in [80], there is no known strongly-polynomial time algorithm to solve them.

Here, we try to make a more effective use of DP-based methods by obtaining structural

\(^{1}\text{This chapter is based on our publication in [22].}\)
results for optimal control problems. In structural results, a DP formulation is used to characterize the structure of possible optimal policies. Then, knowledge of the policy structure can be exploited to design very efficient numerical methods to find the optimal policy. As an example, in [10], it is shown that the optimal control policy for a single cellular Base-Station (BS) is the well-known guard-channel policy. Then, knowing that the guard-channel policy is fully determined by a single threshold, the authors of [10] propose an efficient method to find it.

We limit our focus to a two-tier wireless network architecture. An example of such a network is depicted in Fig. 5.1. With some modifications, the analysis presented here can be applied to more complex scenarios. We prove that for this architecture the optimal policy has a two-dimensional threshold structure. Moreover, this structural result is used to design two computationally efficient algorithms, Structured Value Iteration (SVI) and Structured Update Value Iteration (SUVI). These algorithms can be used to determine the optimal policy in terms of thresholds. Although the first algorithm is closer in its operation to the conventional Value Iteration (VI), the second algorithm has a significantly lower complexity. Extensive numerical observations suggest that, for all practical parameter sets, the algorithms always converge to the overall optimal policy.
The rest of the chapter is organized as follows. In Section 5.2, the system model and assumptions are presented. Section 5.3 presents the structural results and discussion on the complexity of algorithms to solve optimal CAC problems. In Section 5.4, the proposed algorithms are explained, and numerical results are given in section 5.5, followed by chapter summary in Section 5.6.

5.2 Network Model

A HWN can possibly have a complex configuration, involving many different wireless service layers. It is generally difficult to analytically tract such complicated scenarios to provide insight into the optimal control of resources in HWNs. In this work, we assume a two-tier heterogeneous wireless network architecture consisting of an overlay and an underlay. This basic 2-tier entity will be called a Cluster. An example cluster is shown in Fig. 5.1. There are also neighboring clusters from which horizontal handovers are possible to this cluster. We assume tight coupling between different layers of wireless network [36] in a cluster. In the tight coupling architecture, the management of different layers is centralized. In what follows, we assume that there exists a control unit which makes the CAC decision for the underlay and overlay BSs, and that clusters act independently and can measure the rate of external arrival processes such as hand-overs from neighbor clusters. Note that our mathematical analysis and control algorithms are independent of underlying wireless technologies as long as they satisfy some general technical requirements. However, in the simulation section parameters are chosen with respect to IEEE 802.16 WiMAX and IEEE 802.11 WLAN standards.

Service requests (more specifically calls in this work) arrive according to a memoryless Poisson process, and also service times are memoryless. Average service times are $\mu_c$ and $\mu_w$ for calls inside overlay and underlay. We also assume a memoryless mobility pattern where calls move to neighbor clusters or different layers at exponentially distributed times with rates given in Table 5.1. It is clear that these assumptions result in exponential channel holding
This is an essential requirement in the application of MDP methods. In this chapter, we focus on call-level QoS, which is common in CAC literature. Moreover, the fixed channel allocation (FCA) scheme is used and $C_c$ and $C_w$ denote the capacity of overlay and underlay in terms of the maximum number of calls they can accommodate. FCA easily applies to various wireless technologies with channel being frequency, time-slot or code assignment. Based on the results in [82] and [83], we consider the case where the number of available voice/multimedia channels in underlay/overlay can be quantized.

In our event-based DP, we associate costs to undesirable control decision events. These costs correspond to the dropping or blocking of arriving calls, and they are incurred when a call admission request is rejected by the cluster control unit. They reflect the degradation in the QoS from the service provider’s perspective or the inconvenience of service denial perceived by users. The call and handoff arrival rates and their corresponding rejection costs are shown in Table 5.1. Throughout the rest of this chapter, every call type is called a class.

In the study of CAC schemes several optimality criteria are considered. The most common ones are minimization of a total cost (objective) function and minimization of the blocking probability given some hard constraints on dropping probabilities. In [10], these are refereed to as MINOBJ and MINBLOCK, respectively. The main advantage of MINBLOCK lies in the fact that it can guarantee some upper bounds on the dropping probabilities. This can also be achieved by MINOBJ by adjusting cost ratios. Furthermore, MINBLOCK has the drawback of not taking into account how much resource is wasted in reservation to achieve those bounds [46]. In this chapter, we focus on MINOBJ for its flexibility. We can formally define MINOBJ as

$$\text{MINOBJ} : \quad \min g_{\pi} = \sum_{k=1}^{L} C^{(k)}_R \lambda_k P_B^{(k)}$$  \hspace{1cm} (5.1)$$

where $C^{(k)}_R$ is the cost of rejecting a call request of class $k$, $\lambda_k$ is the arrival rate of class $k$ calls, $P_B^{(k)}$ is the blocking (dropping) probability for that class and $L$ is the total number of call classes.
<table>
<thead>
<tr>
<th>#</th>
<th>Rate</th>
<th>Rejection Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\lambda_c$</td>
<td>$C_{NBC}$</td>
<td>New calls to Overlay</td>
</tr>
<tr>
<td>2</td>
<td>$\lambda_w$</td>
<td>$C_{NBW}$</td>
<td>New calls to Underlay</td>
</tr>
<tr>
<td>3</td>
<td>$\eta_{hcc}$</td>
<td>$C_{HDCC}$</td>
<td>Handoff to Overlay from Overlay</td>
</tr>
<tr>
<td>4</td>
<td>$\eta_{hcw}$</td>
<td>$C_{HDWC}$</td>
<td>Handoff to Underlay from Overlay</td>
</tr>
<tr>
<td>5</td>
<td>$\eta_{hwc}$</td>
<td>$C_{HDWC}$</td>
<td>Handoff to Underlay from Overlay</td>
</tr>
</tbody>
</table>

Table 5.1: Call Arrival/Handoff Rates and Rejection Costs.

### 5.3 Optimal CAC Policy

Decision theoretic optimization for Markovian processes is a well-known stochastic control method [45]. The Markov property allows significant reduction in tabular programming complexity and in some cases makes it possible to obtain structural results. An MDP is determined by four components: state space $S$, action space $A$, state transition probabilities $P(.)$, and a cost function $C(.)$. The performance criteria can be formulated with respect to finite or infinite horizons and for average-cost or discounted-cost problems. The solution to an MDP is called a policy or rule. A policy maps the state space to actions $\pi : S \rightarrow A$, such that the optimization goal is achieved. A large class of policies, in which the decision is independent of time, are called stationary policies.

In this work, we wish to minimize the average expected cost for an infinite-horizon problem. This reflects our concern about long-run QoS performance. We start with a finite-horizon optimal cost function and we show that the solution to the infinite-horizon problem has the same structure. Let us denote by $V_k(i, j)$ the optimal cost function for a $k$-stage problem with the initial state $(i, j)$ where $i$ is the number of calls in overlay and $j$ is the number of calls in underlay at the start of the decision epoch. Using the uniformization technique [71], we can
write $V_{k+1}(i, j)$ recursively as

$$V_{k+1}(i, j) = \frac{\lambda_c}{v_{\text{max}}} \min(V_k(i, j) + C_{NBC}, V_k(i + 1, j))$$

$$+ \frac{\lambda_w}{v_{\text{max}}} \min(V_k(i, j) + C_{NBW}, V_k(i, j + 1))$$

$$+ \frac{\lambda_{hc}}{v_{\text{max}}} \min(V_k(i, j) + C_{HDC}, V_k(i + 1, j))$$

$$+ \frac{i\eta_{hcc}}{v_{\text{max}}} \min(V_k(i - 1, j) + C_{HDCW}, V_k(i, j + 1))$$

$$+ \frac{j\eta_{hcc}}{v_{\text{max}}} \min(V_k(i, j - 1) + C_{HDCC}, V_k(i + 1, j - 1))$$

$$+ \frac{i\mu_c}{v_{\text{max}}} V_k(i - 1, j) + \frac{jw}{v_{\text{max}}} V_k(i, j - 1)$$

$$+ \frac{\lambda_{out}^{hc}}{v_{\text{max}}} V_k(i - 1, j) + (1 - \frac{v_{\text{out}}(i, j)}{v_{\text{max}}}) V_k(i, j)$$

(5.2)

where $v_{\text{out}}(i, j)$ is the rate of going out of state $s = (i, j)$,

$$v_{\text{out}}(i, j) = \lambda_c + \lambda_w + \lambda_{in}^{hc} + \lambda_{out}^{hc} + i\eta_{hcc}$$

$$+ j\eta_{hcc} + i\mu_c + jw,$$  

(5.3)

$\lambda_{out}^{hc} = i\eta_{hcc}$, and $v_{\text{max}}$ is the uniformization parameter such that $v_{\text{max}} \geq v_{\text{out}}(i, j)$ for every $(i, j)$ pair. Since $v_{\text{out}}(i, j)$ is increasing in $i$ and $j$, we choose $v_{\text{max}} = v_{\text{out}}(C_c, C_w)$. Equation (5.2) consists of nine terms, each reflecting one possible event; the first three terms reflect arrivals to the cluster, the fourth and fifth terms account for vertical handovers, the next three terms are for departure events and the last term is due to the uniformization technique where staying in the same state is possible. We also assume the following boundary conditions

$$V_k(C_c + 1, j) = \infty \quad \text{and} \quad V_k(-1, j) = 0 \quad 0 \leq j \leq C_w$$

$$V_k(i, C_w + 1) = \infty \quad \text{and} \quad V_k(-1, j) = 0 \quad 0 \leq i \leq C_c.$$  

(5.4)

### 5.3.1 Optimality of Threshold-Based Policy

We show that the optimal policy to minimize the average cost for the system model given in Section 6.3 is a 2D threshold-based policy. In a single threshold system, that threshold is
independent of the system state. When the system state is more complex, such as in the HWNs case, the threshold for the operation of one system component might depend on the state of another one. In our scenario, it gives rise to a 2D threshold structure.

From (5.2), it can be seen that when a call of class \( L \) arrives, it is only admitted if \( V_k(i', j') - V_k(i, j) \leq C^L_R \), where state \( s = (i, j) \) is the current state, state \( t = (i', j') \) is the next state if we admit the call, and \( C^L_R \) is the rejection cost for class \( L \). Let us define two difference operators for \( V_k(i, j) \),

\[
\Delta^i V_k(i, j) = V_k(i, j) - V_k(i-1, j)
\]

\[
\Delta^j V_k(i, j) = V_k(i, j) - V_k(i, j-1).
\] (5.5)

For every fixed \( j \) there is a sequence of \( \Delta^i V_k(i, j) \) for \( i = 1 \ldots C_c \), and vice versa. In what follows we claim that the sequences of \( \Delta^i V_k(i, j) \) and \( \Delta^j V_k(i, j) \) are increasing in \( i \) and \( j \), respectively.

**Lemma 5.3.1.** \( V_k(i, j) \) is convex and monotonically non-decreasing in \( i \) (or \( j \)) for every fixed \( j \) (or \( i \)). (Proof is given in Appendix A)

It has been shown in [71] that for average-cost problems with finite \( S \) and \( A \), the optimal policy is stationary. Moreover, we are only interested in stationary policies which result in irreducible chains. The chain defined by \( V_k(i, j) \) is also aperiodic since it contains loops into the same state. According to Theorem (6.6.2) in [71], for irreducible and aperiodic markov decision processes the difference of upper and lower bounds of \( V_{k+1}(i, j) - V_k(i, j) \) converges to the optimal average cost per unit time when \( k \to \infty \). Also, Theorem (6.6.1) in [71] implies that the optimal per-unit-time average cost function has the same structure as \( V_k(i, j) \) defined in (5.2). Hence, structural results on \( V_k(i, j) \) would directly hold for the infinite-horizon per-unit-time cost function.

**Theorem 5.3.1.** A 2D threshold-based policy is an optimal solution to the control problem with the system model given in (5.2).
Proof. Without loss of generality, let us assume that a call of class L arrives to overlay when the system state is $s = (i - 1, j_0)$. The proof for arrivals to underlay is similar. If the call is admitted, increase in the optimal cost function is $\Delta V_k(i, j_0)$. We show that the CAC decision can be expressed in terms of thresholds determined by $\Delta V_k(i, j_0)$ and $C_{LR}^L$.

From Lemma 5.3.1, we know that the sequence of $\Delta V_k(i, j_0)$ is increasing in $i$. If there is an $\hat{i}$ for which $\Delta V_k(\hat{i}, j_0) \leq C_{LR}^L$ and $\Delta V_k(\hat{i} + 1, j_0) > C_{LR}^L$, then $\hat{i}$ is the threshold for admission to overlay when there are $j_0$ calls in underlay. Otherwise, if for every $\hat{i} \in \{1, \ldots, C_c\}$ we have that $\Delta V_k(\hat{i}, j_0) \leq C_{LR}^L$ then that call is of high priority and it is only rejected when the system is full. Also, if for every $\hat{i}$, $\Delta V_{j_0}(\hat{i}, j) > C_{LR}^L$ then that call class is of low priority and it is never admitted to the system.

Note that for every call class of $L$, threshold $\hat{i}$ depends on $\Delta V_k(\hat{i}, j_0)$ which in turn depends on $j_0$. This implies that the threshold for overlay operation depends on the underlay state. Therefore, the optimal control policy has to be 2D threshold-based to account for this correlation.

\[ \begin{align*} \end{align*} \]

5.3.2 CAC Algorithm

We denote by $\pi = \langle \vec{T}_c[C_w, M], \vec{T}_w[C_c, N] \rangle$ the class of threshold-based polices. Here, $M$ is the number of call classes entering overlay and $N$ is the number of call classes entering underlay. Every class within $M$ or $N$ would be called a subclass. In our scenario $M$ is 3 and $N$ is 2. The CAC algorithm when system state is $s = (i, j)$ at the arrival epoch and policy $\pi$ is employed is given in Algorithm 4. When a call of subclass $L'$ arrives to overlay (underlay), it is only admitted if the number of active calls in overlay (underlay) is less than the threshold for that call-type. This threshold is a function of call subclass and number of calls in the other network underlay (overlay).

A CAC algorithm is fully determined given policy $\pi$ in terms of thresholds. However, finding these values is a non-trivial problem. Efficient computation of these thresholds is considered in the next section.
Algorithm 2 2D Threshold-Based CAC

Input: $\pi = (\overrightarrow{T_c}[C_w, M], \overrightarrow{T_w}[C_c, N])$

A Call of class $L$ arrives

It belongs to subclass $L'$

Output: Admission Decision

1: if Arrival to overlay then
2: \hspace{1em} if $i < \overrightarrow{T_c}(j, L')$ then
3: \hspace{1.2em} return Admit
4: \hspace{1em} else
5: \hspace{1.2em} return Reject
6: \hspace{1em} end if
7: else \{Arrival to underlay\}
8: \hspace{1em} if $j < \overrightarrow{T_w}(i, L')$ then
9: \hspace{1.2em} return Admit
10: \hspace{1em} else
11: \hspace{1.2em} return Reject
12: \hspace{1em} end if
13: end if
5.3.3 Finding Policy $\pi$

A major requirement for CAC algorithms is their adaptivity to network traffic dynamics. Since this is generally achieved by periodically updating the admission policy, the algorithm computational load has to be minimal. Depending on the system size, the computation cost of solving a general MDP can be very high. Several methods such as Value Iteration (VI), Policy Iteration (PI) and Linear Programming (LP) methods are developed to solve general MDP problems [14].

According to [80], no strongly-polynomial algorithm is known for solving MDPs. Although MDPs can be solved by conversion to LP problems, polynomial-time algorithms for LP are inefficient and impractical. On the other hand, practical LP algorithms can result in exponential time-complexity in the worst case when used to solve MDPs. Consequently, there are no efficient and practical polynomial-time algorithms to solve MDPs. Therefore, the computation cost of finding thresholds for the optimal policy can be a burden if we use any of these techniques. However, when we know about the optimal solution structure, we might be able to exploit this knowledge to solve the problem more efficiently.

Generally, either direct or indirect methods can be employed to find the CAC parameters, i.e., policy thresholds. Direct methods require calculating the average cost for a given policy $\pi_1$. This can be done by modeling the system as a continuous markov chain (CTMC). Note that every MDP problem given a policy $\pi_1$ can be analyzed as a Markov chain. Then Gaussian elimination-like methods can be used to find state probabilities and to calculate the average cost. Once we have the average cost we can use methods such as multidimensional bisection search [84] to find the parameters that minimize it. The problem with this method is that for a two-tier network each having capacity $n$, the size of Markov chain state space would be $O(n^2)$ and Gaussian elimination would take $O((n^2)^3) = O(n^6)$.

However, as explained previously, CAC algorithms have to be light weight to be of any practical use. In indirect methods we avoid a direct evaluation of cost function. Instead we use an iterative approximation. Along with that, we use our prior knowledge of optimal policy
structure to further improve the algorithm time-complexity.

5.4 Efficient Computational Algorithms

In this section, we introduce efficient computational algorithms to find the optimal CAC policy. We first describe the conventional Value Iteration (VI) algorithm. We then propose two efficient algorithms called Structured Value Iteration (SVI) and Structured Update Value Iteration (SUVI). The basic principle of these algorithms is similar to VI. However, we use our prior knowledge of the optimal solution structure to eliminate unnecessary computations.

5.4.1 Conventional VI Algorithm

Conventional Value Iteration (VI) algorithm is based on the Bellman-Ford iterative equation [14],

\[ V_n(s) = \min_{a \in A(s)} \{ c_s(a) + \sum_{t \in S} P_{st}(a)V_{n-1}(t) \}. \]  

Note that this equation is backward in time, such that \( V_0(.) \) is the cost at the last time step. In every iteration \( V_n(s) \) is calculated for all \( s \in S \). Here, \( S \) is the state space, and \( A(s) \) is the set of possible actions at state \( s \). \( P_{st}(a) \) is the transition probability of going from \( s \) to \( t \) having taken action \( a \), and \( c_s(a) \) is the cost of taking action \( a \) in state \( s \). In our model, the system state has two components, the number of calls in overlay \( i \) and the number of calls in underlay \( j \); \( s = (i, j) \). For every incoming call, either new or hand-off, at any state two actions are possible: accept (denoted by 1) or reject (denoted by 0); \( A(s) = \{0, 1\} \).

To find the state transition probabilities \( P_{st}(a) \), we use fictitious decision epochs [71]. The computational complexity of evaluating the cost function in every step highly depends on the density of the \( P_{st}(a) \) matrix. When times between decision epochs are exponentially distributed we can reduce the computation cost by introducing fictitious decision epochs at which no real decision has to be made. These correspond to departure events when no action is taken. By this
technique, at every decision epoch either real or fictitious, the system state can only change to adjacent states, making many terms in $P_{st}(a)$ zero. However, to keep track of the epoch type we have to extend the state space by one dimension. The increased computation cost due to this enlarged state-space is compensated by the reduction in the $P_{st}(a)$ density.

We define the new state variable to be a triple $s = (i, j, k)$. Here, $k$ is the departure or arrival type. We already have 5 call types from Table 5.1. We add a fictitious call event type of 0 which corresponds to call departures with a fictitious decision of $a = 0$ to be taken at departure events. In addition, since the decision epochs can be at any randomly distributed time, a Semi-Markov Decision Process (SMDP) model need to be used [71]. Again, we take the uniformization rate to be $v_{max} = v_{out}(C_c, C_w)$. Also, we have to determine $v_s(a)$, the rate of going out of state $s$ having taken action $a$. Here we give the transition probabilities for some of the state-action combinations in terms of transition rates with $P_{st}(a) = \frac{q_{st}(a)}{v_s(a)}$ and $s = (i, j, k)$:

$$
q_{st}(a = 1) = \begin{cases}
(i + 1)(\eta_{hec} + \mu_c) & t = (i, j, 0) \\
 j\mu_w & t = (i + 1, j - 1, 0) \\
 \lambda_c & t = (i + 1, j, 1) \\
 \lambda_w & t = (i + 1, j, 2) \\
 \lambda_{in}^{hec} & t = (i + 1, j, 3) \\
 (i + 1)\eta_{hec} & t = (i + 1, j, 4) \\
 j\eta_{hec} & t = (i + 1, j, 5)
\end{cases} \quad (5.7)
$$

for $k \in \{1, 3\}$ and $v_s(a = 1) = v_{out}(i + 1, j)$ and $v_{out}(i, j)$ given in (5.3). Another example for
$k = 4$ is

$$
q_{st}(a = 0) = \begin{cases}
(i - 1)(\eta_{hcc} + \mu_c) & t = (i - 2, j, 0) \\
(j + 1)\mu_w & t = (i - 1, j, 0) \\
\lambda_c & t = (i - 1, j, 1) \\
\lambda_w & t = (i - 1, j, 2) \\
\lambda_{in}^{hcc} & t = (i - 1, j, 3) \\
(i - 1)\eta_{hcc} & t = (i - 1, j, 4) \\
(j + 1)\eta_{hwc} & t = (i - 1, j, 5)
\end{cases}
$$

(5.8)

with $v_s(a) = v_{out}(i - 1, j)$. Note that in the above, a hand-off request from overlay to underlay was initially rejected ($a = 0$) leaving only $i - 1$ calls in overlay at the start of the decision epoch. The transition probability would have been as follows if we were to admit that call $s = (i, j, 4)$:

$$
q_{st}(a = 1) = \begin{cases}
(i - 1)(\eta_{hcc} + \mu_c) & t = (i - 2, j + 1, 0) \\
(j + 1)\mu_w & t = (i - 1, j, 0) \\
\lambda_c & t = (i - 1, j + 1, 1) \\
\lambda_w & t = (i - 1, j + 1, 2) \\
\lambda_{in}^{hcc} & t = (i - 1, j + 1, 3) \\
(i - 1)\eta_{hcc} & t = (i - 1, j + 1, 4) \\
(j + 1)\eta_{hwc} & t = (i - 1, j + 1, 5)
\end{cases}
$$

(5.9)

and $v_s(a) = v_{out}(i - 1, j + 1)$.

We specify the boundary conditions by defining

$$
V_n(i, C + 1, j, k) = \infty \text{ for all } j \text{ and } k \\
V_n(C, j + 1, k) = \infty \text{ for all } i \text{ and } k \\
V_n(-1, j, k) = 0 \text{ for all } j \text{ and } k \\
V_n(i, -1, k) = 0 \text{ for all } i \text{ and } k
$$

(5.10)

For SMDP, (5.6) needs to be modified to reflect the semi-Markov state transition rates. We
define operator $T_V[V(\cdot), s, a]$ as

$$
T_V[V(\cdot), s, a] = c_s(a)v_s(a) + \frac{v_s(a)}{v_{\text{max}}} \sum_{t \in S} P_{st}(a)V(t) + \left(1 - \frac{v_s(a)}{v_{\text{max}}} \right)V(s).
$$

(5.11)

Given this operator we can rewrite (5.6) for SMDPs as

$$
V_n(s) = \min_{a \in A(s)} \{T_V[V_{n-1}, s, a]\}.
$$

(5.12)

5.4.2 SVI Algorithm

![Figure 5.2: SVI algorithm operation; AR, BR and RR for calls coming to underlay and $D = 1$.](image)

Theorem 5.3.1 states that the optimal solution is a 2D threshold policy, implying that the admission region for any call type should be a closed area. An example of this is shown in Fig. 5.2. For any given policy $\pi_1$ and call subclass, we can partition the state space into three disjoint areas, called Accept-Region (AR), Border-Region (BR), and Reject-Region (RR). We
define the region indicator function \( I_R(s, p) \) for state \( s = (i, j) \) and call request of subclass \( p \) as

\[
I_R(s, p) = \begin{cases} 
\text{AR} & i - \bar{T}_c(j, p) < -D \\
\text{BR} & \left| i - \bar{T}_c(j, p) \right| \leq D \\
\text{RR} & i - \bar{T}_c(j, p) > D.
\end{cases}
\]

(5.13)

If a state is within distance \( D \) of the threshold level then it is in \( \text{BR} \). \( D \) acts as a tuning parameter, determining the size of area we are willing to re-evaluate in every iteration. An example of \( I_R(s, p) \) classification is shown in Fig. 5.2 for \( D = 1 \), where dotted states correspond to the threshold levels. The indicator function for the underlay subclasses is similar. Given the indicator function \( I_R(s, p) \), we can redefine the action space \( A(s) \) as \( A'(s) \),

\[
A'(s) = \begin{cases} 
\{0\} & \text{if } I_R(s, p) = \text{RR} \\
\{1\} & \text{if } I_R(s, p) = \text{AR} \\
\{0, 1\} & \text{if } I_R(s, p) = \text{BR}.
\end{cases}
\]

(5.14)

Here, we are limiting the set of possible actions. The idea is that for states inside the admission region it would be unnecessary to consider a possible reject action if they are not close to the border. Note that the cost function evaluation is done iteratively until an optimum policy is reached. Thus, it can be expected that if an optimum action is not currently taken in a state, the algorithm will eventually reach that state and would choose the appropriate action. As we will see in the next section, extensive numerical observations suggest that, for all practical parameter sets under consideration, the algorithm always converges to the overall optimal policy.

Finally, the proposed SVI algorithm is given in Algorithm 5 as an extension to VI [71]. As in the VI algorithm, \( \epsilon \) determines the desired accuracy. Note that in Algorithm 5 the new action space \( A'(s) \) is used to improve the algorithm efficiency.
Algorithm 3 Structured Value Iteration Algorithm

**Input:** \( \pi, P_{st}, c_s(a), D, v_{max}, \epsilon \)

**Output:** Optimal Admission Policy

1: **Initialize** \( V_0(s) \)
   
   s.t. \( \forall s \in S : 0 \leq V_0(s) \leq \min_a\{c_s(a)v_s(a)\} \)
   
   \( n := 0 \)
   
   \( \forall s \quad \pi_0(s) = 1 \)

2: \( n := n + 1 \)

3: **Find** \( \forall s \in S \)

   \( V_n(s) = \min_{a \in A'(s)} \{TV[V_{n-1}, s, a]\} \)
   
   \( \pi_n(s) = \arg \min_{a \in A'(s)} \{TV[V_{n-1}, s, a]\} \)

4: **Compute**

   \( m_n = \min_{t \in S} \{V_n(t) - V_{n-1}(t)\} \)
   
   \( M_n = \max_{t \in S} \{V_n(t) - V_{n-1}(t)\} \)

5: \( \forall j, l \quad \overrightarrow{T}_{c}^n[j, l] = \arg \min_i \{\pi_n(i, j, l) = 0\} \)

   \( \forall i, l \quad \overrightarrow{T}_{w}^n[i, l] = \arg \min_j \{\pi_n(i, j, l) = 0\} \)

   Recompute \( A'(s) \) based on \( \overrightarrow{T}_{c}^n \) and \( \overrightarrow{T}_{w}^n \)

6: **if** \( M_n - m_n \leq \epsilon m_n \) **then**

7: Policy \( \pi_n \) is optimal and \( \overrightarrow{T}^n \) is the threshold set

8: **Stop**

9: **else**

10: **Go to** step 2

11: **end if**
5.4.3 SUVI Algorithm

The intuition behind SVI was that it is not necessary at every point to evaluate the optimal cost function for all possible actions when some actions are very unlikely to be the optimal decision. In SVI, we assign a default action to every such point based on the region it belongs to, and then in every round of iteration the cost function for that point is updated according to that default action. However, we believe that some of these default updates might be non-necessary if there has not been a major decision or cost change in their neighborhood. Also, under the operation of a numerical algorithm similar to SVI, in every iteration the changes in cost or decisions can only happen within the border region. Therefore, it might be more efficient if we limit the scope of the cost function update to a neighborhood of the border region. More formally, let us define the Update Region (UR) indicator function $I_U(s, p)$ as

$$I_U(s, p) = \begin{cases} 
1 & |i - T_{e(j, p)}| \leq D_u \\
0 & \text{otherwise}
\end{cases} \quad (5.15)$$

A state belongs to the UR if it is within distance $D_u$ ($D_u \geq D$) of the threshold levels. An example of $I_U(s, p)$ classification is shown in Fig. 5.3 for $D = 1$ and $D_u = 2$. We define the action space $A''(s)$ as

$$A''(s) = \begin{cases} 
\{0\} & \text{if } I_R(s, p) = RR \text{ and } I_U(s, p) = 1 \\
\{1\} & \text{if } I_R(s, p) = AR \text{ and } I_U(s, p) = 1 \\
\{0, 1\} & \text{if } I_R(s, p) = BR \\
\emptyset & \text{otherwise}
\end{cases} \quad (5.16)$$

The algorithm proposed for SVI in the last section can be used in conjunction with this new action space $A''(s)$ to find the optimal policy.

5.5 Numerical Results

The performance of the proposed methods is studied in this section. We use a combination of a C++ discrete event simulator and a MATLAB numerical program for that purpose. First, an
optimal policy is found through iterative numerical calculations in MATLAB. Then, it is fed into the discrete-event simulator to find the resultant QoS performance. All parameters take their default values shown in Table 6.2 unless otherwise stated. For the rest of this section, we define the rejection cost vector \( \overline{C}_R = \{C_R^{(1)}, \ldots, C_R^{(5)}\} \), where \( C_R^{(k)} \) is the rejection cost for call requests of class \( k \) as classified in Table 5.1.

### 5.5.1 The Optimal Policy

The quality of the control policies obtained by using the proposed algorithms and their abilities to achieve the optimal performance is considered in this section. We conducted an extensive number of experiments to compare the result of SVI and SUVI with conventional VI. All comparisons of the threshold levels obtained by using SVI/SUVI and VI showed complete matching. These observations suggest that, for all practical parameter sets, the algorithms always converge to the overall optimal policy. Hence, they can be considered as reliable methods to
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_c$</td>
<td>50 calls</td>
<td>$\mu_c$</td>
<td>0.01 sec$^{-1}$</td>
</tr>
<tr>
<td>$C_w$</td>
<td>20 calls</td>
<td>$\mu_w$</td>
<td>0.01 sec$^{-1}$</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>0.5 calls/sec</td>
<td>$C_{NBC}$</td>
<td>5</td>
</tr>
<tr>
<td>$\lambda_w$</td>
<td>0.1 calls/sec</td>
<td>$C_{NBW}$</td>
<td>3</td>
</tr>
<tr>
<td>$\eta_{hcc}$</td>
<td>5 × 10$^{-3}$ sec$^{-1}$</td>
<td>$C_{HDC}$</td>
<td>50</td>
</tr>
<tr>
<td>$\eta_{hcw}$</td>
<td>0.01 sec$^{-1}$</td>
<td>$C_{HDCW}$</td>
<td>10</td>
</tr>
<tr>
<td>$\eta_{hwc}$</td>
<td>0.02 sec$^{-1}$</td>
<td>$C_{HDWC}$</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 5.2: Arrival Rates and Rejection Costs Used in Simulation.

perform CAC for HWNs.

Figure 5.4 illustrates the ARRs for the optimal policy obtained from SVI. It is assumed that $C_c = 50$ and $C_w = 20$. The ARRs for call class of 2 (new arrival to underlay) are depicted. Figure 5.5 shows the same results for another configuration in which the mobility rate $\eta_{hwc}$ has increased from 0.02 to 0.08. Comparison of these figures shows that at the lower mobility rate, the rejection of calls is more evenly distributed over the set of states. Also, it can be observed that in both figures nearly the same percentage of states are in the rejection area. This is due to the fact that the prioritization of calls in SVI is mostly based on cost ratios defined in Table 5.1, rather than the mobility pattern.

5.5.2 Convergence Speed

In section 5.3, we explained that the complexity of CAC algorithms can impose limitations on their practicality. We compare the convergence speed of SVI and SUVI against VI for different
values of $D$ and $D_u$. Figures 5.6 and 5.7 show the Time Gain, $T_{Gain}^X = TVI / TX$, for SVI and SUVI over VI for a range of network capacities. The run time of each algorithm is obtained by using MATLAB to record its CPU time. In this experiment, $C_c$ varies from 20 to 100 while maintaining $C_w = \frac{1}{3} C_c$. For every $C_c$, $\lambda_c$ is chosen such that the nominal overlay utilization, $\rho_c = \frac{\lambda_c}{C_c \mu_c}$, is equal to 1, and $\lambda_w = 0.2 \times \lambda_c$. Also, we assume $\epsilon = 10^{-3}$.

As the network capacity increases, the ratio of BR area, in which full optimization is performed, relative to the area of AR/RR regions, in which a default action is evaluated, decreases. This accounts for the increased efficiency with the increase in network capacity. Figure 5.7 shows $T_{Gain}$ for the SUVI algorithm. Each curve gives the time gain of the SUVI over VI for different network capacities for a given $D$ and $D_u$. It can be observed that smart selection of update points can significantly improve the convergence speed. It is interesting to note that the best-performing parameters are $D = 1$ and $D_u = 2$, while the algorithm always converges to the optimal policy regardless of the selected tuning parameters.
5.5.3 Optimal Policy vs. Complete Sharing (CS)

A Complete Sharing (CS) policy refers to the admission policy in which \( \pi(s) = 1 \), regardless of the system state as long as it remains within system capacity boundaries; in our scenario \( s = (i, j) \leq (C_c, C_w) \). In what follows, we compare the performance of the optimal policy obtained from SUVI (with \( D = 1 \) and \( D_u = 2 \)) with the CS policy. CS policy essentially represents the performance of a system in which no proactive resource allocation scheme is employed and call request are satisfied as long as there are enough resources. Comparing the optimal policy with the CS policy shows how effective the control algorithm is and what degrees of performance gain can be expected by adopting a non-trivial policy.

Figure 5.8 shows the result of this comparison for two different cost vectors, \( \mathcal{C}_R \), when \( \lambda_c \) varies from 0.5 to 1.0. \( \lambda_c \) is chosen such that the nominal overlay utilization, \( \rho_c \), ranges from 1.0 to 2.0. Also, we have \( C_c = 50, C_w = 20 \) and \( \lambda_w = 0.2 \times \lambda_c \). In this experiment, we consider two cases having different cost vectors of \( \mathcal{C}_{R1} = \{5, 3, 50, 10, 30\} \) and \( \mathcal{C}_{R2} = \{5, 3, 20, 10, 20\} \). First, the optimal policies, \( \pi_1 \) and \( \pi_2 \), are computed for each cost vector for the given system parameters, and then a discrete event simulation is performed to obtain the
resultant average cost $g_\pi$. The system is then exposed to the same load/cost settings under a CS policy to find the corresponding $g_{cs}$. Note that $g_\pi = \sum_{k=1}^{5} C_R^{(k)} \lambda_k P_B^{(k)}$.

Several insights can be gained from this figure. Let us define the cost improvement difference as $Q(\lambda) = g_{cs}(\lambda) - g_\pi(\lambda)$ and the cost reduction as $Y(\lambda) = \frac{g_{cs}(\lambda)}{g_\pi(\lambda)}$. First, $Q(\lambda)$ invariably increases when system is exposed to higher loads, which implies the control algorithm is more useful at higher loads. An interesting observation is that the cost reduction, $Y(\lambda)$, is higher for $C_R^{1}$ compared to the cost reduction for $C_R^{2}$. This is due to the fact that $C_R^{1}$ assigns a more diverse set of rejection costs to call classes. The closer the rejection costs are, the lower the cost reduction is going to be, and in the extreme case of having equal rejection costs, i.e., $C_R^{(k)} = C_0$ for $k = 1, \ldots, 5$, both CS and optimal policy yield the same average cost and $Y(\lambda)$ will have its minimum value of 1.

Figure 5.9 shows the comparison results for the case when the area ratio $R_a$ of the underlay to overlay increases from 0.05 to 0.5. Let us assume that the total arrival rate to the cluster, consisting of overlay and underlay, is $\lambda_T = \lambda_c + \lambda_w$, and depending on the area’s expanse
covered by each layer, a fraction $R_a$ of $\lambda_T$ is assigned to the underlay while $1 - R_a$ goes to the overlay. Under user assignment strategies in which a user is initially assigned to the underlay if it is within the double-coverage area (overlap of underlay and overlay), $R_a$ is directly proportional to the underlay expanse. Two cases are considered in this experiment. Policy $\pi_3$ is obtained for $\lambda_T = 0.5$ and $\pi_4$ is obtained for $\lambda_T = 1$, and also $\overline{C}_R = \{5, 3, 20, 20, 20\}$.

It can be observed from Fig. 5.9 that the cost difference $Q(R_a)$ is higher when the load is higher. One notable observation is that the average cost for the optimal policy $\pi_4$ decreases when $R_a$ increases. This can be associated with the lower rejection cost for new arrivals to the underlay. When $R_a$ increases, a larger number of incoming calls are assigned to underlay which if rejected accumulate a lower cost as compared to their rejection cost at the overlay. The non-monotonic trend in the average cost for CS policy $\pi_{CS4}$ can be explained similarly.

Another set of average cost results is shown in Fig. 5.10. The cost ratio $R_c$ used in this figure is defined as $R_c = \frac{CHDCW}{CNBW}$. Also, the rejection cost vector is $\overline{C}_{R1} = \{5, 5, 20, 5 \times R_c, 20\}$. Policy $\pi_5$ is computed for $\lambda_c = 0.5$ and policy $\pi_6$ is computed for $\lambda_c = 1$, while maintaining

Figure 5.7: SUVI convergence speed gain $T_{Gain}$ for $D = 1, 2, 3$ and $D_u = 2, 3, 4$. 
Figure 5.8: Optimal and CS policy costs versus incoming traffic.

Figure 5.9: Optimal and CS policy costs versus underlay area.
Figure 5.10: Optimal and CS policy costs versus rejection cost ratio.

\( \lambda_w = 0.2 \times \lambda_c \). Again, the higher the load, the larger is the difference between the average cost induced by the optimal and CS policies. A trend in this figure which requires explanation is the difference in rates at which average costs for the optimal and CS policies increases when \( R_c \) increases. In contrary to the expectation that the cost reduction \( Y(R_c) \) should have remained fixed, as it was the case in Fig. 5.8, we see that \( Y(R_c) \) is increasing in \( R_c \). The reason is that when we increase \( R_c \) the blocking/dropping probabilities which constitute \( g_\pi \) remain constant under the CS policy, and hence, the average cost grows proportionally, whereas under the optimal policy these probabilities are adaptively changing to achieve the minimum possible average cost for any given \( R_c \). The overall result is that the optimal control scheme does not allow for a linear change in system rejection costs to be reflected severely in the average cost.

### 5.5.4 QoS Performance

In this section, the control policy generated by SUVI is used to find the resultant performance. We use the policy obtained in the last section for \( \eta_{hwc} = 0.02 \). The system is exposed to
an increased mobility rate $\eta_{hwc}$ to study its response. Figure 5.11 shows the results. It can be observed that the dropping probabilities for calls of classes 3 and 4, $P_{DCC}$ and $P_{DWC}$, are increasing. The increase of total incoming load to overlay has also elevated the blocking probability for new arrivals $P_{BC}$. More interestingly, the change in $P_{DCW}$ is not monotonic, and it initially decreases and then increases. The initial decrease happens when the increase of $\eta_{hwc}$ results in lowered levels of load in the underlay while overlay load has not significantly changed. The increasing part can be correlated to the situation in which many calls in overlay, including previous handovers from underlay, try to handover to the underlay.

Figure 5.11: Dropping/blocking probabilities for increased mobility.
5.6 Chapter Summary

In this chapter, optimal CAC for HWN is considered. Structural results on the optimal cost function for a two-tier HWN architecture are presented. It is shown that for such networks the cost function is convex. This is used to prove that the optimal CAC policy is two-dimensional threshold based. Then, efficient numerical methods called Structured Value Iteration (SVI) and Structured Update Value Iteration (SUVI) are proposed to determine the optimal admission policy. Although the first one is closer in its operation to the conventional Value Iteration algorithm, the second one has a significantly lower complexity.

Extensive simulation and numerical studies show that SVI and SUVI are both reliable and effective. Firstly, they always converge to the optimal policy in much less time compared to conventional numerical methods used to solve MDPs. Also, discrete-event call-level simulations confirm that the obtained policy is effective in maintaining the desired QoS performance.

The proposed method is a new framework for systems with complex state descriptions. It can potentially be used in stochastic control of queuing systems. In general, under some minor technical assumptions, the same method can be applied to many problems involving optimal control for stochastic systems with multidimensional state-space.
5.7 Nomenclature
### Table 5.3: Symbols used and their description.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu_c$</td>
<td>Service rate in cellular network</td>
</tr>
<tr>
<td>$\mu_w$</td>
<td>Service rate in WLAN network</td>
</tr>
<tr>
<td>$C_c$</td>
<td>Capacity of overlay cellular BS</td>
</tr>
<tr>
<td>$C_w$</td>
<td>Capacity of underlay mesh AP</td>
</tr>
<tr>
<td>$g_\pi$</td>
<td>System average cost (infinite-horizon) for policy $\pi$</td>
</tr>
<tr>
<td>$P_B^{(k)}$</td>
<td>Blocking probability for calls of class $k$</td>
</tr>
<tr>
<td>$C_R^{(k)}$</td>
<td>Rejection cost for calls of class $k$</td>
</tr>
<tr>
<td>$L$</td>
<td>Total number of call classes</td>
</tr>
<tr>
<td>$V_k(i,j)$</td>
<td>Optimal cost for a $k$-stage problem with initial state $(i,j)$</td>
</tr>
<tr>
<td>$v_{\max}$</td>
<td>Uniformization rate</td>
</tr>
<tr>
<td>$v_{out}(i,j)$</td>
<td>The rate of leaving state $(i,j)$</td>
</tr>
<tr>
<td>$\lambda_{hcc}^{out}$</td>
<td>Handoff departure rate to neighboring overlay cells</td>
</tr>
<tr>
<td>$\lambda_{hcc}^{in}$</td>
<td>Handoff arrival rate from neighboring overlay cells</td>
</tr>
<tr>
<td>$\lambda_c$</td>
<td>Arrival rate of new calls to overlay</td>
</tr>
<tr>
<td>$\lambda_w$</td>
<td>Arrival rate of new calls to WLAN</td>
</tr>
<tr>
<td>$\eta_{hwc}$</td>
<td>Handoff rate from WLAN to to overlay</td>
</tr>
<tr>
<td>$\eta_{hcw}$</td>
<td>Handoff rate from overlay to WLAN</td>
</tr>
<tr>
<td>$\eta_{hec}$</td>
<td>Handoff rate from overlay to neighboring cells</td>
</tr>
<tr>
<td>$q_{st}(a)$</td>
<td>Transition rate from state $s$ to $t$ having taken action $a$</td>
</tr>
</tbody>
</table>
Chapter 6

Overlay and Wireless Mesh Network Integration

6.1 Overview

An appealing networking architecture is the integration of Wireless Mesh Networks (WMN) [85] with the overlaying cellular infrastructure. Mesh networking is used in the underlay to expand network coverage and to increase its capacity. It is expected that in the near future mesh networks will be extensively deployed in urban and suburban areas to provide mobile and roaming users with wireless networking services. An example of such a network is shown in Figure 6.1.

In this chapter, we generalize the result of Chapter 5 to the case in which underlay is a wireless mesh network\(^1\). We consider the characterization of structured optimal CAC policies for HWNs formed by the integration of mesh and cellular networks. Such HWN architectures are promising because mesh networks can be easily deployed to enhance network capacity at a cost lower than that of expanding cellular infrastructure. However, having multiple Access Points (AP) in the mesh underlay network adds one more dimension to the complexity of CAC problem, and user movements between different APs create additional complications in the

\(^1\)This chapter is based on our publication in [24].
problem formulation. To the best of our knowledge, there is no known solution to the optimal admission control problem for this architecture.

By making some practical assumptions regarding the service model provided by this class of HWNs, we develop a scalable approximate model to achieve near-optimal performance in the control of these networks. The core assumption is that the priority in modeling of different layers of a HWN should be given to the provisioning of overflows. A new or a hand-over call overflows to the overlay when it cannot be accommodated by the target mesh Access Point (AP) at the underlay. It is well known that this overflow traffic is not Poisson [54]. In order to characterize the overflow traffic from the underlaying mesh to the overlay, a Partially Observable Markov-Modulated Poisson Process (PO-MMPP) traffic model is developed. This model captures the burstiness of the overflow traffic under the imperfect observability of the mesh network status. Afterwards, the overlay network is modeled as a controlled PO-MMPP/M/C/C queuing system, and structural optimal control results are given. The significance of these results is in showing that the optimal control policies for this class of HWNs can be characterized as monotonic threshold curves.

Moreover, the monotonic threshold curve structure of the optimal CAC policy is used to
design a computationally efficient Structured Coordinate Search Algorithm (SCSA) to determine the optimal policy in terms of thresholds. Based on this algorithm, we propose a modular CAC scheme, which can be employed in cellular base stations and in mesh-controller nodes to improve the system performance. Through numerical analysis and simulations, we show that the proposed algorithm has a convergence speed much faster than that of the Value Iteration (VI) [45] algorithm and leads to a CAC scheme significantly more cost efficient than the Complete Sharing and Guard-Channel policies. We present two versions of SCSA, a base version that is highly efficient, and an extended version, SCSA-OPT, which is guaranteed to converge to the optimal policy at a slightly slower convergence speed. We further discuss the characteristics of these algorithms in Section 6.7.

The rest of the chapter is organized as follows. Related works and their contrasts to our contributions are discussed in Section 6.2. In Section 6.3, the system model and assumptions are discussed. The general framework used in the design of decision-theoretic optimal CAC schemes is explained in Section 6.4. The development of an approximate overflow model for this HWN architecture is given in Section 6.5. In Section 6.6, structural results for the control of overlay network are presented. The algorithm to find the structured optimal control policy are given in Section 6.7. Simulation and numerical results are presented in Section 6.8, followed by chapter summary in Section 6.10.

6.2 Related Work and Research Contribution

To the best of our knowledge heterogeneous networks comprising of IEEE 802.11 mesh and cellular network have not be previously studied. In this chapter, not only we look at this appealing integrated network, but we also characterize the structure of the CAC policies for efficient QoS provisioning. In what follows, we briefly refer to relevant research studies and discuss how they relate to this work.

There are several studies on QoS for HWNs where the conventional guard-channel policy
is used and the available channels are divided between handover and new calls coming from different layers. In [37], admission control for voice and data services in a HWNs is considered and an iterative algorithm is proposed to find the dropping and blocking probabilities. In [43], the minimization of linear cost functions is used to find the number of guard channels.

In all of the aforementioned studies the reservation scheme is chosen a priori, i.e., it is assumed that a fixed threshold policy is capable of achieving the optimal performance. This is in contrast to the approach taken in this thesis in which we do not make any assumption about the structure of the optimal policy, and rather, we try to rigorously characterize it by allowing it to emerge naturally out of our decision theoretic framework.

One of the earliest works on structural results is the stochastic knapsack problem [18]. In [16], event-based dynamic programming is used to formulate call admission control of multiple classes in a resource-sharing system. They show that customer classes might be ordered in some cases. When classes are ordered, if admitting one class is optimal, then admitting a call of a more rewarding class is optimal as well. In a recent work [15], structured CAC policies for resource-sharing systems are studied. The authors consider two major classes of structured policies, namely, reservation and threshold policies, and propose fast search algorithms to find the policy parameters. The problem of structural results for cellular networks has been considered in [10], and it is shown that the the optimal CAC for a single cellular BS has a threshold structure. It is notable that none of the aforementioned studies consider HWNs, or are applicable to the case of HWNs due to prohibitively high complexity.

Another related research direction is the modeling of overflow traffic for hierarchical macro/micro-cellular networks. Different models with various complexities are studied in [58, 59, 61, 64]. These models are built upon prior research on the characterization of overflow processes for loss queuing systems [55, 56, 62, 86]. Initially, the overflow traffic in multilayer wireless networks has been modeled as a Poisson process, but it is known that this approximation is inaccurate, and that by failing to model the burstiness of the overflow traffic, it can underestimate the blocking of overflow calls at the overlay [54]. In contrast, Interrupted Poisson Processes
(IPP) and Markov-Modulated Poisson Processes (MMPP) have been used to better model the behavior of overflow processes in the context of hierarchical cellular networks [58–61].

A common assumption among studies on resource allocation for mobile networks is that the traffic statistics of all micro-cells (or underlay cells) are equal, i.e., they have identical arrival and hand-off rates. This reduces analysis complexity by allowing one to focus on one micro-cell, while assuming similar results for the rest. In contrast, for the model considered in this work, because of the asymmetric structure of mesh networks, it is impractical to assume that traffic arrival and handoff rates are similar at different mesh APs. For example, consider the mesh network depicted in Fig. 6.1. Mesh AP $\kappa$ receives handover arrivals from three neighbors, APs $i$, $j$ and $l$. While Mesh AP $\imath$ only receives handovers from two neighbors, APs $j$ and $\kappa$. We do not require the assumption of having identical traffic statistics at every mesh AP. Potentially, this can result in intractable complexity. In this work, we propose a method to analyze and control this non-uniform system at a reduced complexity level.

In [22], we have studied a simple two-tier HWN with a single underlay network (e.g., a single wireless LAN) and have shown that the optimal CAC policy has a two-dimensional threshold structure. However, such results are not directly applicable to HWNs with a more complex underlay architecture such as mesh networks. This is mainly due to the exponential growth of the system state space with respect to the number of mesh APs. In this work, through a novel modeling of the overflow stream, we devise a control policy whose state space increases linearly in size as the number of Mesh APs increases.

### 6.3 System Model

#### 6.3.1 Network Model and Underlying Processes

In this work, we focus on a two-tier HWN architecture consisting of an overlay and an underlay, where the underlay can be a mesh network with multiple wireless APs. This basic two-tier entity will be called a *cluster*. An example cluster is shown in Fig. 6.1. New service requests
(more specifically calls in this work) are assumed to arrive according to a memoryless Poisson process. Call holding times are exponentially distributed with averages of $\mu_c$ and $\mu_w$ for calls at the overlay and underlay, respectively. We also assume a memoryless mobility pattern where calls move to neighbor clusters or different layers at exponentially distributed times with rates as denoted in Table 6.9. Here, $\eta_{hw}^{i,j}$ denotes the rate of handover for a call from AP $i$ to AP $j$ as depicted in Fig. 6.1. The following relations hold between parameters defined in Table 6.9:

$$\eta_{hw}^{i,j} = \phi(i, j)\eta_h^i$$ and $$\eta_{hwc}^i = \phi(i, 0)\eta_h^i$$, where index 0 is reserved for cellular overlay. It is clear that the above memoryless assumptions result in exponential channel holding times [81].

In this chapter, we focus on call-level QoS, which is common in CAC literature. Moreover, the fixed channel allocation (FCA) scheme is used. FCA applies to various wireless technologies with channel being frequency, time-slot or code assignment. Let $C_c$ and $C_w$ denote the capacity of overlay BS and underlay mesh APs measured in basic bandwidth units (BBU). We assume that all multimedia calls require a single BBU. This corresponds to the single service system studied in [15].

Heterogeneous wireless networks are expected to accommodate a combination of QoS-sensitive and best-effort data services. In this work, we focus on multimedia services that require access to communications resources in a guaranteed fashion. Data services, if they require guaranteed bandwidth, can be modeled along with multimedia calls, and if they are best-effort services, will utilize the available unused bandwidth at any time instant, and as such do not effect our analysis.

To accommodate the increasing demand for best-effort data services future wireless networks have to become more elastic in their bandwidth management. This can be utilized to provide delay and resource sensitive applications with better QoS through dynamic readjustment of allocated bandwidth. However, in this work we are interested in studying the behavior of systems under tight resource availability. Systems running on excess capacity, which can easily accommodate best-effort and QoS-sensitive services, are not the target of this study.
6.3.2 Call Handling Policy

In [22], tight coupling between different layers of wireless networks [36] in a cluster, in which management of different layers is centralized, was assumed. However, when the underlay is comprised of multiple APs, it is impractical to assume centralized control, because the network management traffic overhead will be excessively high. In what follows, we assume that there exists a control unit that makes the CAC decision for calls coming to the overlay based on the state of the overlay and some partial knowledge about the status of the underlay mesh network. This partial observation of the mesh status is such that overlay is made aware of dropping/blocking APs due to overflow new and handover calls, but otherwise has no additional information on the state of all mesh APs. This is an essential requirement for a scalable and practical CAC scheme.

We assume that once a call overflows to overlay, it stays at that layer until completion, i.e., repacking is not allowed. This common assumption allows one to analyze the underlay independently of the overlay network [59, 61]. Note that the no-repacking policy serves as an indirect scheme for classification and layer assignment of users with different velocities. In particular, calls made by a highly mobile user is more likely to request an overflow and hence more likely to be eventually assigned to the overlay during the call period, which clearly is a desirable outcome for system efficiency. Our proposed model can be extended to consider the effect of repacking on system performance. This can be done by assuming that a certain fraction of calls in overlay, depending on the network topology, will be repacked to underlay APs. Clearly, allowing repacking introduces a loop in the system analysis through mutual dependence of traffic rates at overlay and underlay. This will significantly increase the time complexity of methods used to calculate system performance.

When a new call request arrives that is within the double coverage area (DBCA), it will first seek admission to the underlay network. If it can not be accommodated by the underlay, it overflows to the overlay network. Also, when a handover call moves from an underlay AP to either the single coverage area (SCA), or another underlay AP where it can not be
accommodated, it overflows to the overlay network. For a new call, if it is initiated within the SCA, it will directly seek admission to the overlay.

6.3.3 Underlay Mesh Network Operation

We assume that the capacity offered by every mesh node to its local users is fixed. As no repacking of overlay calls is allowed, mesh APs receive no traffic load from the overlay, and hence, for the resource reservation in every mesh AP, we only need to take into account its local traffic and the load coming from its immediate neighbors in the mesh topology. For this purpose, any of the previously studied CAC schemes can be used [9, 10].

6.4 Optimization Framework

Dynamic programming optimization using Markov Decision Processes (MDPs) is a well-known discrete-time stochastic control method [45]. An MDP is determined by four components: state space $S$, action space $A$, state transition probabilities $P$ (or transition rates $Q$ in case of continuous-time processes), and a cost function $C$. The solution to an MDP is called a policy or rule. A policy maps the state space to actions $\Psi : S \rightarrow A$, such that the optimization goal is achieved. A large class of policies, in which the decision is independent of time given the system state, is called stationary policy.

In our event-based DP, we associate costs to undesirable control decision events. These costs correspond to the dropping or blocking of incoming calls. A list of rejection costs for different call types is given in Table 6.1. Note that the cost of dropping an AP-to-AP handover is not considered, since a rejected AP-to-AP handover call simply creates an overflow handover to the overlay. Throughout the rest of this chapter, every call type is called a class.

In the study of CAC schemes, the most common performance criterion is the minimization of a total cost function which is referred to as MINOBJ in [10]. We formally define MINOBJ
### Rate Rejection Cost Description

<table>
<thead>
<tr>
<th>Rate</th>
<th>Rejection Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{nc}$</td>
<td>$C_{BNC}$</td>
<td>Blocking of new calls</td>
</tr>
<tr>
<td>$\lambda_{no}^i$</td>
<td>$C_{BNO}$</td>
<td>Blocking of new call overflows</td>
</tr>
<tr>
<td>$\lambda_{hec}^m$</td>
<td>$C_{DCC}$</td>
<td>Dropping of cell-to-cell handovers</td>
</tr>
<tr>
<td>$\lambda_{ho}^i$</td>
<td>$C_{DHO}$</td>
<td>Dropping of handover overflows</td>
</tr>
</tbody>
</table>

Table 6.1: Rejection Costs for New and Handoff calls.

The MDP performance criteria (cost function) can be formulated with respect to finite or infinite horizons, and for average-cost or discounted-cost problems. Our objective is to minimize the average cost per unit time for an infinite-horizon non-discounted problem. This reflects our concern about long-run QoS performance. In addition, because the decision epochs can be at any randomly distributed time, a Semi-Markov Decision Process (SMDP) model is used [71]. The system state space depends on the information available to the CAC unit. In the ideal scenario where all information are available, one might define the system state space $S$ as the set of all vectors containing the numbers of occupied channels at the overlay BS and at the mesh APs. However, this would lead to a prohibitively large state space. In practice, however, the CAC unit is not likely to obtain complete information about the underlay APs. In Section 6.5, we discuss how to characterize the SMDP for the CAC model described in Section 6.3, by analyzing the underlay overflow into the overlay.

In what follows, we present a general framework that will be used throughout the rest of the chapter. We start with a finite-horizon optimal cost function, and we show that the solution as

$$\text{MINOBJ : } \min_{\psi} g_{\psi} = \sum_{l=1}^{L} C_{R}^{(l)} \lambda_l P_{R}^{(l)}$$

where $C_{R}^{(l)}$ is the cost of rejecting a call request of class $l$, $\lambda_l$ is the arrival rate of class $l$ calls, $P_{R}^{(l)}$ is the rejection (either blocking or dropping) probability for that class and $L$ is the total number of call classes.
to the infinite-horizon problem has the same structure. Let us denote $V_k(s)$ to be the minimum expected cost function for a $k$-stage problem with the initial state $s$. And let $E_A$ denote the set of arrival events that are controllable and can potentially be rejected at the costs defined in Table 6.1, and let $E_N$ denote the set of internal transitions and departures for which no action needs to be taken. Using the uniformization technique [71], we can write $V_{k+1}(s)$ recursively as

$$V_{k+1}(s) = \frac{1}{v_{\text{max}}} \left\{ \sum_{e \in E_A} q_e \left[ \min\{\Delta V_k(\psi_e s), C_{R(e)}\} + V_k(s) \right] + \sum_{e \in E_N} q_e V_k(\psi_e s) + (v_{\text{max}} - v_{\text{out}}(s)) V_k(s) \right\}$$  

(6.2)

where $q_e$ is the rate of transition for events of type $e$, $C_{R(e)}$ is the cost of blocking event $e$, $v_{\text{out}}(s)$ is the rate of going out of state $s$, and $v_{\text{max}}$ is the uniformization parameter such that $v_{\text{max}} \geq v_{\text{out}}(s)$ for every $s$. Here, operator $\psi_e$ acts on state $s$ and returns the resultant state if event $e$ was to be admitted. The $\Delta$ symbol for an operator $\psi$ is defined as

$$\Delta V_k(\psi_e s) = V_k(\psi_e s) - V_k(s).$$  

(6.3)

Equation (6.2) consists of three terms, each reflecting one possible event. The first term account for arrivals to the cluster, and the second term accounts for departures and internal non-controllable transitions. The last term is due to the uniformization technique where staying in the same state is possible. Note that overflow events, despite being internal are controllable, and as such belong to $E_A$. In the first term, $\Delta V_k(\psi_e s)$ is the cost of admitting a call whose arrival is triggered by event $e$. If this cost is less than the blocking cost $C_{R(e)}$, then the call will be admitted; otherwise it is rejected. More general information on similar frameworks can be found in [10, 75].

It is well known that for average-cost problems with finite $S$ and $A$, and time-invariant transition probabilities, the optimal policy is stationary. Moreover, we are only interested in stationary policies that result in irreducible chains. For such irreducible and aperiodic MDPs, the upper and lower bounds of $V_{k+1}(s) - V_k(s)$ converge to the optimal average cost per unit
time when \( k \to \infty \) [71]. More formally, given

\[
M_k = \max_{t \in S} \{ V_{k+1}(t) - V_k(t) \},
\]

\[
m_k = \min_{t \in S} \{ V_{k+1}(t) - V_k(t) \},
\]

the optimal cost \( g_{\Psi} \) as defined in (6.1) is equal to

\[
g_{\Psi} = \lim_{k \to \infty} M_k = \lim_{k \to \infty} m_k.
\]

The structural results for \( V_k(s) \) as defined in (6.2) will hold for the optimal per-unit-time average cost function if the underlying Markov decision process is irreducible and aperiodic ([71], Theorem (6.6.1)). The implications of this is that the structure of the optimal policy to achieve MINOBJ is the same as the policy which minimizes \( V_k(s) \).

### 6.5 Analysis Model for Underlay Overflow

In this section, we develop an analytical framework to characterize the overflow traffic from the underlay mesh into the overlay cell. This will provide a foundation for establishing the state transition probabilities in (6.2), allowing further analysis of the solution structure of optimal CAC in Section 6.6.

#### 6.5.1 Analysis of Mesh Underlay

A simplifying assumption commonly employed in the analysis of micro-cellular networks is to equate outgoing and incoming handoff rates for every micro-cell [61]. In our model, due to the asymmetric nature of the mesh network, we can not use such simplifications.

Let \( \lambda_T^i \) denote the total traffic arrival to underlay AP \( i \); we have

\[
\lambda_T^i = \lambda_{nw}^i + \sum_{v=1,v \neq i}^{M} \phi(v,i) \lambda_{hw}^v,
\]

where \( \lambda_{nw}^i \) denotes arrival rate of new calls to underlay AP \( i \), \( \lambda_{hw}^v \) represents the total rate of handoff out of AP \( v \) and \( \phi(v,i) \) denotes the fraction of calls going from AP \( v \) to AP \( i \). For AP \( i \), we define the utilization factor as \( \rho_i = \lambda_T^i / (\mu_w + \eta_i^h) \) and \( \delta_i = \lambda_{nw}^i / \lambda_T^i \). As discussed in Section 6.3, the CAC scheme used in underlay network can be chosen arbitrarily from available
schemes such as a guard-channel policy [9, 10]. For a guard-channel CAC policy with \( g \) guard channels the blocking performance of AP \( i \) can be found, using an extension of Erlang-B formula, as [10]

\[
P_{m_i}(k) = \frac{\rho_i^k \prod_{j=1}^{k} \gamma_j}{k! \sum_{l=0}^{C_w} (\rho_i^l \prod_{j=1}^{l} \gamma_j / l!)}
\]

\[
P_B^{(i)} = \sum_{k=C_w-g}^{C_w} P_{m_i}(k)
\]

\[
P_D^{(i)} = P_{m_i}(C_w),
\]

(6.6)

where \( \gamma_j = 1 - \delta_j I_{\{j \geq C_w-g\}} \) and the indicator function \( I_{\theta} \) equals 1, only if \( \theta \) is true. \( P_{m_i}(k) \) is the probability of having \( k \) active calls in AP \( i \). When \( g = 0 \), that is when no control is exercised to give priority to handover calls, (6.6) reduces to the well-known Erlang-B loss formula.

The probability that a call in AP \( i \) will require a handover before termination is \( R_h^{(i)} = \eta^i_h / (\eta^i_h + \mu_w) \). For the total rate of calls leaving AP \( i \) due to handover, we have

\[
\lambda_{hw}^i = R_h^{(i)} \left\{ (1 - P_D^{(i)}) \sum_{v=1, v \neq i}^{M} \phi(v, i) \lambda_{hw}^v \right. \\
+ \left. (1 - P_B^{(i)}) \lambda_{nw}^i \right\}.
\]

(6.7)

This forms a set of non-linear equations, with solution being the set of \( \lambda_{hw}^i \) for all APs. Due to possible existence of cycles in the network topology, calculation of \( P_B^{(i)}, P_D^{(i)} \) and \( \lambda_T \) may involve iterations. For example, the Erlang Fixed-Point approximation has been used to find the traffic intensities and blocking probabilities in loss networks [86, 87]. We start with a set of values for \( P_B^{(i)} \) and \( P_D^{(i)} \), and solve (6.7) to find the set of \( \lambda_{hw}^i \). Then, (6.6) is used to recompute a new set of \( P_B^{(i)} \) and \( P_D^{(i)} \). This iteration is repeated until a desired accuracy is reached. The proof for convergence of this iterative technique to the proper solution, and the uniqueness of the result is given in [86].

In the rest of this chapter, for ease of illustration, we assume that the underlay does not distinguish between new and handover calls (i.e., guard-channel policy with \( g = 0 \), which implies that \( P_D^{(i)} = P_B^{(i)} \)). It is notable that the framework presented below can be easily extended to accommodate a wide set of underlay CAC schemes.


6.5.2 Analysis of Overflow Traffic

Three events trigger the overflow of a call from an underlay AP to overlay: 1) when an active call (mobile station) associated with an AP leaves the double-coverage area (DBCA), 2) when a new call arrives to a blocking AP, and 3) when a handover call comes to a dropping (fully utilized) AP. The overflow generated by events of the first type form a Poisson stream with a total rate of

\[ \lambda_{po} = \sum_{v=1}^{M} \phi(v, 0) \lambda_{hw}^v. \]  

(6.8)

However, for events of type 2 and 3, the overflow is not uniformly distributed over all AP states, and as studied previously, it has a bursty nature [55, 58, 59, 61]. The Markov Modulated Poisson Process (MMPP) can be used to model such overflow streams. The overflow modeling is depicted in Fig. 6.2. If AP \( i \) is in the blocking states, e.g., \( m_i \in \{C_w - g, \ldots, C_w\} \), new calls coming with rate \( \lambda_{no}^i = \lambda_{nw}^i \), will overflow. If the AP is in dropping mode, e.g., \( m_i = C_w \), then in addition to new calls, incoming handovers will overflow too. The rate of handover overflow from AP \( i \) when it is in the dropping mode is

\[ \lambda_{ho}^i = \sum_{v=1,v \neq i}^{M} \phi(v, i) \lambda_{hw}^v. \]  

(6.9)

An MMPP\((Q, \Lambda)\) with \( J \) states is identified by two parameters, \( Q \) a \( J \times J \) matrix, and \( \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_J) \), a diagonal matrix of size \( J \times J \). \( Q \) is the infinitesimal generator for the MMPP underlying process, and \( \Lambda \) determines the rate generated at each state. Let

---

**Figure 6.2: Underlay overflow model.**

Overflow Stream \( \text{MMPP}(\widehat{Q}_v, \widehat{\Lambda}_v) \)

Underlay

Mesh

AP

Overflow Stream

Mesh

Underlay

AP

Overflow Stream

Figure 6.2: Underlay overflow model.
MMPP\((Q_i, \Lambda_i)\) denote the model for the bursty overflow from AP \(i\). In our example \(g = 0\), traffic overflows with rate \(\lambda_{ho}^i + \lambda_{no}^i\) whenever the system is at state \(m_i = C_w\). At other states the overflow rate is zero, i.e., \(\Lambda_i = \text{diag}(0, \ldots, 0, \lambda_{ho}^i + \lambda_{no}^i, 0, \ldots, 0)\).

To simplify the aggregate overflow characterization, we further approximate each of these MMPPs by an IPP. An IPP\((\lambda, \alpha, \beta)\) has three parameters. It is either in state ON or in state OFF. If it is in state ON, it generates a Poisson stream with rate \(\lambda\). \(1/\alpha\) is the mean of on-time period, and \(1/\beta\) is the mean of off-time period. An IPP can be expressed as an MMPP with two states and parameters

\[
Q_{IPP} = \begin{bmatrix} -\beta & \beta \\ \alpha & -\alpha \end{bmatrix}, \quad \Lambda_{IPP} = \begin{bmatrix} 0 & 0 \\ 0 & \lambda \end{bmatrix}.
\]

(6.10)

To find the equivalent IPP of an MMPP, we employ the technique used in [62]. The method matches the first two non-central moments of the arrival rate of the MMPP. Let \(\pi_{Q_i}\) denote the stationary probability vector associated with \(Q_i\) such that \(\pi_{Q_i} Q_i = 0\) and \(\pi_{Q_i} e = 1\). The \(n\)th non-central moment of the instantaneous rate of MMPP\((Q_i, \Lambda_i)\) is \(m^{(n)} = \pi_{Q_i} \Lambda_i^n e\). Now, the parameters of equivalent IPP\((\lambda_i, \alpha_i, \beta_i)\), for AP \(i\), can be determined as

\[
\lambda_i = \lambda_{ho}^i + \lambda_{no}^i,
\alpha_i = [1 - \pi_{Q_i}(C_w)]/\tau_c,
\beta_i = \pi_{Q_i}(C_w)/\tau_c,
\]

(6.11)

where \(v\) is variance of instantaneous rate given as \(v = m^{(2)} - (m^{(1)})^2\), and time constant, \(\tau_c\), as

\[
\tau_c = v^{-1} \left[ \pi_{Q_i} \Lambda_i (e \pi_{Q_i} - Q_i)^{-1} \Lambda_i e - (m^{(1)})^2 \right].
\]

(6.12)

The total overflow from underlay can be modeled as the superposition of the above IPPs. In what follows, we assume that these IPPs are statistically independent. This assumption, which helps by making the mathematical formulation tractable, is commonly used in the study of overflow traffic as reviewed in Section 2.4. We take a close look at this assumption and its potential inaccuracies in Section 6.9.
If we cast the $i$th IPP as a two state MMPP as given in (6.10) with $(\tilde{Q}_i, \tilde{\Lambda}_i)$, the aggregate model can be represented by an MMPP $(Q_T, \Lambda_T)$ with

$$Q_T = \tilde{Q}_1 \oplus \tilde{Q}_2 \oplus \cdots \oplus \tilde{Q}_M$$

$$\Lambda_T = \lambda_{po}I + \tilde{\Lambda}_1 \oplus \tilde{\Lambda}_2 \oplus \cdots \oplus \tilde{\Lambda}_M$$  \hspace{1cm} (6.13)

where operator $\oplus$ is the Kronecker sum [62]. Kronecker sum is defined as

$$A \oplus B = A \otimes I_b + I_a \otimes B,$$ \hspace{1cm} (6.14)

and Kronecker product $\otimes$ is such that given an $m \times n$ matrix of $A$ and a $p \times q$ matrix $B$, their product, $C = A \otimes B$, is an $(mp) \times (nq)$ matrix with block elements $c_{ij} = A_{ij}B$. Here is an example of how Kronecker sum operates on matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \oplus \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & b_{12} & a_{12} & 0 \\ b_{21} & a_{11} + b_{22} & 0 & a_{12} \\ a_{21} & 0 & a_{22} + b_{11} & b_{12} \\ 0 & a_{21} & b_{21} & a_{11} + b_{22} \end{bmatrix}$$ \hspace{1cm} (6.15)

$Q_T$, as given in Equation 6.13, has the size of $2^M \times 2^M$; two states for every IPP being ON or OFF. Note that if the underlay APs had similar statistics, as assumed in almost all previous studies, this MMPP could trivially be simplified to an MMPP with $M+1$ states. However, this is not the case for the network model considered in this work due to asymmetries in mesh networks.

The large size of the system state space as shown above would render an exact control algorithm prohibitively complex for practical applications. For example, to fully utilize $Q_T$, the system would need, at every decision epoch, the exact status of every underlay AP, which clearly is not a scalable approach.

In what follows, we adopt a state space reduction approach, to consider only the total number of dropping/blocking APs. In practice, this may correspond to an overlay controller that has only a partial view of the underlay mesh network, such that it is made aware of dropping/blocking APs due to overflow new and handover calls, but otherwise has no additional
information on the state of all APs. Formally, if the state of the $i$th IPP is denoted by $x_i$ (with $x_i = 1$ when IPP is ON), then the underlay state can be expressed as $\vec{u} = (x_1, x_2, \ldots, x_M)$. We define the degree operator on $\vec{u}$ as $\omega(\vec{u}) = \sum_{i=1}^{M} x_i$. At state $\vec{u}$, the overlay controller only knows the total count of ON IPPs, which we denote by random variable $N = \omega(\vec{u})$.

We define $U = \{ \vec{u} | x_i \in \{0, 1\}, \ i = 1, \ldots, M \}$ as the state space of the underlay network. $U$ can be partitioned into $M+1$ disjoint and exhaustive subsets $U(n)$ based on our observation of $N=n$, $n=0, 1, \ldots, M$. We have

$$\lambda_v^{(n)} = E[\lambda_{\vec{u}}|\vec{u} \in U(n)] = \sum_{\vec{u} \in U(n)} \lambda_{\vec{u}} \frac{\pi_t(\vec{u})}{\pi_{U(n)}}. \tag{6.17}$$

However, we need to determine what fraction of $\lambda_v^{(n)}$ is new calls and what fraction is formed by handover calls, as these streams have different rejection costs. Let us denote by $\lambda_{ho}(\vec{u})$, the total rate of handover overflows and $\lambda_{no}(\vec{u})$ the total rate of new calls overflow at state $\vec{u}$. We have

$$\lambda_{ho}(\vec{u}) = \lambda_{po} + \sum_{i=1}^{M} x_i \lambda_{ho}^i, \quad \lambda_{no}(\vec{u}) = \sum_{i=1}^{M} x_i \lambda_{no}^i. \tag{6.18}$$
Figure 6.3: PO-MMPP overflow model; in state \( n \) two overflow streams are generated; \( \lambda^{(n)}_{hv} \) and \( \lambda^{(n)}_{nv} \).

The total overflow rates for new and handover calls for when PO-MMPP(\( \widehat{Q}_{v}, \widehat{\Lambda}_{v} \)) is in state \( n \) can now be calculated as

\[
\lambda^{(n)}_{hv} = E[\lambda_{ho}(\bar{u})|\bar{u} \in U(n)] = \sum_{\bar{u} \in U(n)} \lambda_{ho}(\bar{u}) \frac{\pi_t(\bar{u})}{\pi_{U(n)}}
\]

(6.19)

\[
\lambda^{(n)}_{nv} = E[\lambda_{no}(\bar{u})|\bar{u} \in U(n)] = \sum_{\bar{u} \in U(n)} \lambda_{no}(\bar{u}) \frac{\pi_t(\bar{u})}{\pi_{U(n)}}
\]

(6.20)

To find birth-death rates, we use the Markov chain state aggregation method [88]. The transition rate from subset \( U(i) \) to \( U(j) \) denoted by \( q_v(i, j) \) can be found as

\[
q_v(i, j) = \sum_{\bar{x} \in U(i)} \sum_{\bar{y} \in U(j)} q_t(\bar{x}, \bar{y}) \frac{\pi_t(\bar{x})}{\pi_{U(i)}}
\]

(6.21)

where \( Q_T = [q_t(\bar{x}, \bar{y})] \). The birth-death rates can be calculated as

\[
\alpha^{(n)}_v = q_v(n - 1, n), \quad \beta^{(n)}_v = q_v(n, n - 1).
\]

(6.22)

Equations (6.17) and (6.22) completely determine the underlay overflow process as PO-MMPP(\( \widehat{Q}_{v}, \widehat{\Lambda}_{v} \)). The resultant MMPP model is depicted in Fig. 6.3.

Here we add two remarks. First, in the modeling of the overflow process, we could go directly from the original MMPPs to PO-MMPP without intermediate conversion to IPPs. However, the computation complexity of finding PO-MMPP transition rates from the initial MMPP model would be too high as it requires operations on excessively large matrices (\( C^w_M \times C^w_M \) versus \( 2^M \times 2^M \)). The second point is that making admission decisions based on observations of \( N \) is not the same as assuming a number of identical and homogeneous APs. The heterogeneous traffic pattern, is reflected in non-linear increases of \( \lambda^{(n)}_{nv} \) and \( \lambda^{(n)}_{hv} \) depicted in Fig. 6.3.
Also, the transition rates of $\alpha^{(n)}_v$ and $\beta^{(n)}_v$ are not simple multiplications of $N$ and a base rate for identical APs.

### 6.6 Structured Control of Overlay BS

In this section, we study structural results for optimal control of the overlay cellular BS and model the overlay network as a controlled PO-MMPP/M/C/C queuing system.

In our system model, controlled transitions at the overlay are 1) new call arrivals to overlay, 2) new call overflows from underlay, 3) handover call arrivals to overlay from neighbor clusters, and 4) handover call overflows from underlay to overlay. With the rejections costs $C_{\text{BNC}}$, $C_{\text{BNO}}$, $C_{\text{DCC}}$ and $C_{\text{DHO}}$, respectively. We refer to each of these request types as a call class. We define the system state as $s = (i, n)$, where $i$ is the number of calls at the overlay and $n$ is the total number of dropping/blocking APs at the underlay. Here, the state space $S$ is defined as

$$S = \{ s = (i, n) : 0 \leq i \leq C_c, 0 \leq n \leq M \}. \quad (6.23)$$

We also define state operators of

$$\mathcal{A} s : s = (i, n) \rightarrow s' = (i + 1, n)$$
$$\mathcal{D} s : s = (i, n) \rightarrow s' = (i - 1, n)$$
$$\mathcal{Q} s : s = (i, n) \rightarrow s' = (i, n + 1)$$
$$\mathcal{R} s : s = (i, n) \rightarrow s' = (i, n - 1). \quad (6.24)$$

These operators are boundary sensitive, which means that they do not map a state to a point outside the system state space. For example, if $s + (0, 1) \notin S$, then $\mathcal{Q} s$ is not a proper operation.

Extending the framework presented in Section 6.4, we can write the optimal cost function
The following Lemmas are required to obtain structural results for $V_k(s)$, where Lemma 6.6.1 is needed to prove Lemma 6.6.2. Proofs are given in the Appendix B.

**Lemma 6.6.1.** $\lambda^{(n)}_{IV}$ and $\lambda^{(n)}_{IV}$ are both monotonically non-decreasing in $n$.

**Lemma 6.6.2.** $V_k(s)$ has the following properties:

A) $\Delta V_k(As) \geq 0$

B) $\Delta V_k(A^2s) \geq \Delta V_k(As)$

C) $\Delta V_k(AQs) \geq \Delta V_k(As)$.

In the above, Property A is needed to prove properties B and C. Intuitively, Property B states that the cost function of the system, and its differences, monotonically increase with the number of calls being accommodated by the system. Property C states that the differences of cost function monotonically increase with the number of dropping/blocking APs at the underlay.

**Definition 1.** A threshold policy is a CAC policy in which resource requests of class $r$ are admitted if and only if the system state is less than the threshold for class $r$, i.e., $s_i < \overline{T}[r]$. 

$$V_k(s)$$ for this system as

$$V_{k+1}(s) = \frac{1}{v_{\text{max}}} \left\{ \lambda^{in}_{hcc} \min[\Delta V_k(As), C_{DCC}] + \lambda^{(n)}_{hv} \min[\Delta V_k(As), C_{DHO}] + \lambda^{(n)}_{IV} \min[\Delta V_k(As), C_{DN}] + \lambda_{nc} \min[\Delta V_k(As), C_{BNC}] + \alpha^{(n)}_v \Delta V_k(Qs) + \beta^{(n)}_v \Delta V_k(Rs) + i(\mu_c + \eta_{hcc}) \Delta V_k(Ds) + v_{\text{max}} V_k(s) \right\}$$

(6.25)

where $v_{\text{out}}(s) = \lambda^{(n)}_{IV} + \lambda^{(n)}_{IV} + \lambda^{in}_{hcc}$

$$v_{\text{out}}(s) = \lambda_{nc} + \lambda^{(n)}_{IV} + \lambda^{(n)}_{IV} + \lambda^{in}_{hcc} + i(\mu_c + \eta_{hcc}) + \alpha^{(n)}_v + \beta^{(n)}_v.$$ 

(6.26)
We denote by $\Pi = \{ \overrightarrow{T} [L, W] \}$ the class of threshold polices. Here, $L = 4$ is the number of controlled call classes entering overlay, and $W$ is the number of possible states the uncontrollable variable can assume. In this scenario, $W = M + 1$. A policy $\Psi \in \Pi$ fully determines the CAC algorithm in terms of thresholds.

**Theorem 6.6.1.** The optimal control policy for the system model given in (6.25) is a threshold policy.

**Proof.** Let us assume that a call of class $l$ arrives to the system with the rejection cost of $C_R^{(l)}$, when the system state is $s = (i, n)$. If the new call is admitted, the increase in the optimal cost function is $\Delta V_k(A(i, n))$. We show that the CAC decision can be expressed in terms of thresholds determined by $\Delta V_k(A(i, n))$ and $C_R^{(l)}$.

If $\exists i_0$ such that $\Delta V_k(A(i_0, n)) < C_R^{(l)}$ and $\Delta V_k(A(i_0 + 1, n)) \geq C_R^{(l)}$, then Lemma 6.6.2.B asserts that for all $i \leq i_0$ we have $\Delta V_k(A(i, n)) < C_R^{(l)}$, and for all $i > i_0$ we have $\Delta V_k(A(i, n)) \geq C_R^{(l)}$. This means that $i_0$ is the state after which the cost of admitting a call of class $l$ becomes more than rejecting it at the cost of $C_R^{(l)}$. Therefore, $\overrightarrow{T}(l, n) = i_0 + 1$. If such $i_0$ does not exist, either for all $0 \leq i \leq C_c$ we have $\Delta V_k(A(i, n)) \geq C_R^{(l)}$ and calls of class $l$ are rejected in all states, or for all $0 \leq i \leq C_c$ we have $\Delta V_k(A(i, n)) < C_R^{(l)}$ and calls of class $l$ are admitted in all states.

In the next theorem, we claim that for every call class, the threshold is a monotonic curve, in that it decreases with an increase in $n$.

**Theorem 6.6.2.** The optimal threshold policy for the system model given in (6.25) belongs to the class of monotonic threshold curves.

**Proof.** We have to show that if $n_2 \leq n_1$, then $\overrightarrow{T}(l, n_1) \leq \overrightarrow{T}(l, n_2)$. Defining $i_1 = \overrightarrow{T}(l, n_1) - 1$ and $i_2 = \overrightarrow{T}(l, n_2) - 1$, we have

\[
\Delta V_k(A(i_1, n_1)) < C_R^{(l)} \leq \Delta V_k(A(i_1 + 1, n_1)) \tag{6.27}
\]

\[
\Delta V_k(A(i_2, n_2)) < C_R^{(l)} \leq \Delta V_k(A(i_2 + 1, n_2)) \tag{6.28}
\]
We want to show that $i_1 \leq i_2$. For contradiction, assume $i_1 > i_2$; then Lemma 6.6.2.B gives us $\Delta V_k(A(i_1, n_2)) \geq \Delta V_k(A(i_2, n_2))$, because $(i_1, n_2) = A^h(i_2, n_2)$ where $h = i_1 - i_2 > 0$. Knowing $i_1 \geq i_2 + 1$ and comparing with (6.28) and applying Lemma 6.6.2.B, we have $\Delta V_k(A(i_1, n_2)) \geq C^{(l)}_R$. Also, from (6.27) we have $\Delta V_k(A(i_1, n_1)) < C^{(l)}_R$. These together give us $\Delta V_k(A(i_1, n_2)) > \Delta V_k(A(i_1, n_1))$. Given Lemma 6.6.2.C, this implies that $n_2 > n_1$ which contradicts our initial assumption.

**Definition 2.** A threshold policy is an ordered threshold policy when we have $\overrightarrow{T}(l, n) \leq \overrightarrow{T}(l', n)$ if $C^{(l)}_R \leq C^{(l')}_R$.

**Theorem 6.6.3.** The monotonic threshold curves from Theorem 6.6.2 are ordered.

**Proof.** We want to show that if $C^{(l)}_R \leq C^{(l')}_R$, then $\overrightarrow{T}(l, n) \leq \overrightarrow{T}(l', n)$. Defining $h = \overrightarrow{T}(l, n) - 1$ and $h' = \overrightarrow{T}(l', n) - 1$, we have

\[
\Delta V_k(A(h, n)) < C^{(l)}_R \leq \Delta V_k(A(h + 1, n))
\]

\[
\Delta V_k(A(h', n)) < C^{(l')}_R \leq \Delta V_k(A(h' + 1, n))
\]

(6.29)

Since $C^{(l)}_R \leq C^{(l')}_R$, from (6.29) we obtain

\[
\Delta V_k(A(h, n)) < C^{(l)}_R \leq C^{(l')}_R \leq \Delta V_k(A(h' + 1, n))
\]

(6.30)

From Lemma 6.6.2.B we know that $\Delta V_k(As)$ is in a non-decreasing function of state $s$. Therefore, we must have $h < h' + 1$ for (6.30) to hold. Considering that thresholds are integer numbers, we can deduce that $h \leq h'$.

In the next section, we use the above structural properties to design an efficient computational algorithm.

### 6.7 Structured Coordinate Search Algorithm

Depending on the system size, computation cost of finding an optimal CAC policy can be very high. The optimal policy can be computed either by using any of Value Iteration (VI), Policy
Iteration (PI), or Linear Programming (LP) techniques, or directly, through explicit formulation of cost criterion and combinatorial search.

In the previous sections, we have derived an approximate overflow model in which partial observation of the underlay network allows a reduction of the problem state space from $C_c C_w M$ to $C_c M$. For this new state space, the size of the search space for optimal control policy is $O(2^{C_c M})$ which is still fairly large. In this section, we exploit our knowledge of the policy structure obtained in the previous section and propose an efficient algorithm which determines the parameters of an optimal threshold policy using combinatorial localized search algorithms [89]. In order to do this, first we have to directly compute $g_{\Psi}$.

Given a CAC policy $\Psi$, the system can be analyzed as a Markov chain with states $(i, n)$ as depicted in Fig. 6.4 to find state probabilities $\pi_s(i, n)$. For transition rates, $q_s$, of this chain we have

$$q_s(i, i + 1) = \lambda_{nc} I_{\mathcal{P}}(i, n, 1) + \lambda_{n}^{(n)} I_{\mathcal{P}}(i, n, 2) + \lambda_{n}^{(n)} I_{\mathcal{P}}(i, n, 3) + \lambda_{n}^{(n)} I_{\mathcal{P}}(i, n, 4)$$

$$q_s(i, i - 1) = i(\mu_c + \eta_{hcc}) \tag{6.31}$$

where call classes $l = 1, 2, 3, 4$ are enumerated according to Table 6.1, and $I_{\mathcal{P}}(i, n, l)$ is the
admission region indicator function given as

\[ I_{\overrightarrow{T}}(i, n, l) = \begin{cases} 
1 & \text{if } i < \overrightarrow{T}(l, n) \\
0 & \text{otherwise.} 
\end{cases} \quad (6.32) \]

Also, as shown in Section 6.5 the PO-MMPP model of the overflow traffic is a birth-death process with rates \( \alpha_v^{(i)} \) and \( \beta_v^{(i)} \), which are given in Equation 6.22. State probabilities \( P(n = k) \) of this chain can be found as

\[ P(n = k) = \frac{\Pi_{i=1}^{k} \alpha_v^{(i)}}{1 + \sum_{j=1}^{M} \Pi_{i=1}^{j} \beta_v^{(i)}}, \quad (6.33) \]

To determine \( g_\Psi \) in (6.1), we have to calculate \( P_R^{(l)} \) and \( \lambda_l \). For call classes associated with \( C_{\text{BNC}} \) and \( C_{\text{DCC}} \) which arrive according to a Poisson process, the rejection probability can be directly computed as

\[ P_R^{(l)} = \sum_{(i, n) \in S} [1 - I_{\overrightarrow{T}}(i, n, l)] \pi_s(i, n). \]

For the overflow processes which arrive according to an MMPP, we have

\[ P_R^{(l)} = \frac{\sum_{(i, n) \in S} \lambda_{hv}^{(n)} [1 - I_{\overrightarrow{T}}(i, n, l)] \pi_s(i, n)}{\sum_{(i, n) \in S} \lambda_{hv}^{(n)} \pi_s(i, n)}, \quad (6.34) \]

Furthermore, overflows \( \lambda_l \) can be calculated as \( \lambda_2 = E_n[\lambda_{hv}^{(n)}] \) and \( \lambda_4 = E_n[\lambda_{hv}^{(n)}] \). We next propose an algorithm to find the optimal \( \Psi^* \) that minimizes \( g_\Psi \).

Without loss of generality, we may sort the the labels of call classes such that if \( l \leq l' \) then \( C_R^{(l)} \leq C_R^{(l')} \). Then, Theorems 6.6.3 asserts that the optimal policy is an ordered monotonic threshold curve. For \( \overrightarrow{T}(l, n) \), this implies the following relations

\[ \text{Monotonic : } \overrightarrow{T}(l, n) \leq \overrightarrow{T}(l, n - 1) \]

\[ \text{Ordered : } \overrightarrow{T}(l, n) \leq \overrightarrow{T}(l + 1, n). \quad (6.35) \]

Let us define the fictitious call classes of 0 (lowest priority) and 5 (highest priority) to set boundary conditions. We have

\[ \overrightarrow{T}(L + 1, n) = C_c, \quad \overrightarrow{T}(0, n) = 0 \quad (6.36) \]

\[ \overrightarrow{T}(l, -1) = C_c, \quad \overrightarrow{T}(l, M + 1) = 0 \quad (6.37) \]
We propose an iterative algorithm that is referred to as the Structured Coordinate Search Algorithm (SCSA). SCSA performs a local search in the state space confined by our structural results. It is given in Algorithm 4. Here, \( g_{\Psi_r}[\vec{T}(l, n) = t] \) is the average cost for policy \( \Psi_r \), except that the value of the threshold at \((l, n)\) is set to \( t \). Also, \( \vec{T}_{\Psi_r}(l, n) \) is the threshold level for a given policy of \( \Psi_r \) at \((l, n)\). In the algorithm, if \( \text{argmin} \) returns more than one threshold, the tie is broken by choosing the smallest one.

\begin{algorithm}
\caption{Structured Coordinate Search Algorithm (SCSA)}
\begin{algorithmic}[1]
\STATE \textbf{Initialize} \( \Psi_0 \) such that: \( \forall (l, n) \quad \vec{T}_{\Psi_0}(l, n) = C_c \)
\STATE \( r := 0 \)
\STATE \( r = r + 1, \Psi_r = \Psi_{r-1} \)
\FOR{\( l := 1, \ldots, L \)}
\FOR{\( n := 0, \ldots, M \)}
\STATE \( b_l = \max \{ \vec{T}_{\Psi_r}(l-1, n), \vec{T}_{\Psi_r}(l, n+1) \} \)
\STATE \( b_u = \min \{ \vec{T}_{\Psi_r}(l+1, n), \vec{T}_{\Psi_r}(l, n-1) \} \)
\STATE \( \vec{T}_{\Psi_r}(l, n) = \text{argmin}_{b_l \leq t \leq b_u} g_{\Psi_r}[\vec{T}(l, n) = t] \)
\ENDFOR
\ENDFOR
\IF{\( g_{\Psi_r} = g_{\Psi_{r-1}} \)}
\STATE Return Policy \( \Psi_r \)
\ELSE
\STATE Go to step 3
\ENDIF
\end{algorithmic}
\end{algorithm}

The algorithm’s operation is based on the fact that the optimal policy has to be an ordered monotonic threshold policy (OMTP) as defined in (6.35). See for example Fig. 6.5, where \( \vec{T}(l, n) \) for \((l, n)\) and its four adjacent points \((l', n') \in \{l-1, l+1\} \times \{n-1, n+1\} \) are depicted. For the OMTP property to hold, \( \vec{T}(l, n) \) cannot assume any value below a lower bound \( b_l = \max \{ \vec{T}(l-1, n), \vec{T}(l, n+1) \} \) or above an upper bound \( b_u = \min \{ \vec{T}(l+1, n), \vec{T}(l, n-1) \} \).

We use this fact to significantly reduce the search space, and at every iteration we greedily
choose the value for $\bar{T}(l, n) \in \{b_l, b_l + 1, \ldots, b_u\}$ that minimizes the total cost.

**Theorem 6.7.1** (Convergence). SCSA converges to an ordered monotonic threshold policy.

**Proof.** We focus on the nested loop given in step 4. The purpose of the loop is to, given an input of $\Psi_{r-1}$, compute a new policy $\Psi_r$. At the start of every iteration of the loop, the loop invariant is that $\Psi_{r-1}$ is an OMTP. To prove that the algorithm yields an OMTP, we have to show that the loop invariant is maintained after each iteration. For $r = 1$, the initial $\Psi_0$ is a complete-sharing policy which is both monotonic and ordered. Within the body of the loop, for each point, the optimal threshold level that minimizes the cost function is found. However, this new threshold is only selected from $\{b_l, \ldots, b_u\}$. This preserves the OMTP property.

To show the convergence of the algorithm, we note that after each iteration there are only two possibilities, either $g_{\Psi_r} < g_{\Psi_{r-1}}$ and this new policy is used in the next iteration, or $g_{\Psi_r} = g_{\Psi_{r-1}}$ and the algorithm terminates. We observe that the total number of policies is $C_{c}^{M,L}$. Therefore, after at most $C_{c}^{M,L}$ steps, the algorithm converges.

Although we know that SCSA converges to an OMTP in a finite time, we do not know if this policy is the globally optimal policy. To shed light on the complexity of this issue, we
discuss several observations. Lemma 6.6.2 states that the k-step optimal cost function is convex with regard to the initial state. We use this in Theorems 6.6.1-6.6.3 to show that the optimal policy is OMTP. SCSA, in turn, exploits this result and finds an OMTP with a minimum cost. The novelty of SCSA is that it searches for the optimal policy knowing not the structure of the cost function \( g_{\Psi_r}(\overrightarrow{T}) \), but rather the structure of the input \( \overrightarrow{T}_{opt} \) that yields the minimum cost.

While searching the state space in SCSA, it is unknown if \( g_{\Psi_r}(\overrightarrow{T}) \) is convex in \( \overrightarrow{T} \). In fact, having known that \( g_{\Psi_r}(\overrightarrow{T}) \) is convex in \( \overrightarrow{T} \) would have allowed us to use steepest descent or multi-dimensional bisection search to find the optimal set of values for \( \overrightarrow{T} \) in the first place. The verification of convexity for \( g_{\Psi_r}(\overrightarrow{T}) \) is particularly challenging given the way its components are specified in (6.34)-(6.34).

In what follows, we propose a second algorithm referred to as the Optimal Structured Coordinate Search Algorithm (SCSA-OPT), which combines SCSA with a policy improvement step to guarantee convergence to the optimal policy. It is well-known that both the PI and VI algorithm converge to the optimal policy starting from any given policy. The idea is that we modify the SCSA stopping rule such that when SCSA converges, we feed the policy it has computed to a modified version of VI to find a better policy. If VI cannot improve this policy anymore, we know that we have found the optimal policy. Otherwise, we feed this newly computed policy, which can be shown to be an OMTP, to SCSA again and repeat the iterations.

More precisely, in the improvement step, when SCSA converges to policy \( \Psi_r \), we compute relative values \( v(s) \) for that policy. Relative values are used in PI and they reflect the cost difference for the long-term average system cost starting from state \( s \) as compared to a reference state \( s_0 \) [71]. From Theorem 6.7.1, we know that \( \Psi_r \) is OMTP and as such \( v(s) \) has to be convex in \( s \). Knowing PI relative values are only a fixed offset away from their VI counterparts for large \( k \), we have \( V_k(s) = k \ g_{\Psi_r} + v(s) \) [71]. \( V_k(s) \) will be convex in \( s \). Then, \( V_k(s) \) is used in (6.25) to compute \( V_{k+1}(s) \) and \( \Psi_{r+1} \). Recall that Lemma 6.6.2 asserts that starting from an initially convex \( V_k(s) \), (6.25) yields a convex \( V_{k+1}(s) \). Therefore, Policy \( \Psi_{r+1} \) is OMTP and can be fed back to SCSA for further improvement. The Policy Improvement (PLI) step is
summarized in Algorithm 5.

**Algorithm 5** Policy Improvement (PLI)

1. **Input:** OMTP Policy $\Psi_r$.
2. Calculate relative values $v(s)$ for $\Psi_r$.
3. Assign $V_k(s) = k g_{\Psi_r} + v(s)$ (for an arbitrary $k$).
4. Use (6.25) to find $\Psi_{r+1}$.
5. Return $\Psi_{r+1}$.

In step 2 of this algorithm where $v(s)$ is calculated, we can use the conventional PI technique of solving the following linear equations

$$v(s) + g_{\Psi_r} = \frac{1}{v_{\text{max}}} \left\{ \lambda_{\text{hcc}}^\in \left[ I_{\bar{T}}(s, 3) \Delta v(A_s) + \bar{T}_{\bar{T}}(s, 3) C_{DCC} \right] \right. + \left. i(\mu_e + \eta_{\text{hcc}}) \Delta v(D_s) + v_{\text{max}} v(s) \right\}$$

$$v(s_0) = 0,$$  \hfill (6.38)

which is similar to (6.25), except that the left-hand side is replaced with $v(s) + g_{\Psi_r}$ and on the right-hand side, $V_k(\psi_es)$ is replaced with $v(\psi_es)$, where $\psi_e$ is the state operator as defined in (6.24). Also, in contrast to (6.25) where $\min$ is used to determine the least cost action, in the above equation, the action is dictated by $\Psi_r$. We use the admission indicator function $I_{\bar{T}}(s, l)$ defined in (6.32) and its complement $\bar{T}_{\bar{T}}(s, l) = 1 - I_{\bar{T}}(s, l)$ to specify this change. Alternatively, we can use iterations to find $v(s)$ as employed in the Modified Policy Iteration [14]. More details on PI relative values formulation can be found in [71]. Also, $s_0$ is an arbitrarily chosen state. In step 3, we choose $k = 0$ so as to avoid the need to recompute $g_{\Psi_r}$. The PLI step is called in Algorithm 6, to check if the SCSA result can be improved by VI.

**Theorem 6.7.2.** SCSA-OPT converges to the optimal policy.

**Proof.** The convergence argument is similar to Theorem 6.7.1. The only notable point is the assignment in step 12 where the result of PLI is assigned to $\Psi_r$. Here, from Lemma 6.6.2 we know that $\text{PLI}(\Psi_r)$ is OMTP, and as such the OMTP loop invariant is maintained even if the
Algorithm 6 Optimal Structured Coordinate Search Algorithm (SCSA-OPT)

1: **Initialize** \( \Psi_0 \) such that: \( \forall(l, n) \quad T_{\Psi_0}(l, n) = C_c \)

2: \( r := 0 \)

3: \( r = r + 1, \Psi_r = \Psi_{r-1} \)

4: **for** \( l := 1, \ldots, L \)

5: **for** \( n := 0, \ldots, M \)

6: \( b_l = \max \left[ T_{\Psi_r}(l - 1, n), T_{\Psi_r}(l, n + 1) \right] \)

7: \( b_u = \min \left[ T_{\Psi_r}(l + 1, n), T_{\Psi_r}(l, n - 1) \right] \)

8: \( \tilde{T}_{\Psi_r}(l, n) = \arg\min_{b_l \leq t \leq b_u} g_{\Psi_r}(T_{\Psi_r}(l, n) = t) \)

9: **if** \( g_{\Psi_r} = g_{\Psi_{r-1}} \) **then**

10: **if** \( \Psi_r = \text{PLI}(\Psi_r) \) **then**

11: Return Policy \( \Psi_r \)

12: **else** \( \Psi_r := \text{PLI}(\Psi_r) \)

13: **end if**

14: **Go to** step 3
policy given by PLI is used. Also, for the policy returned by PLI we have that \( g_{\Psi, t+1} \leq g_{\Psi, t} \), because VI reduces the system cost at every stage and the new policy is at least as good as the previous policy.

This algorithm converges to the globally optimal policy because it only stops when VI cannot improve the policy any further, and we know that VI converges to the optimal policy.

While SCSA-OPT is guaranteed to converge to the optimal policy, its drawback is that it uses VI which is computationally expensive. In our simulation studies, we conducted numerous experiments to compare the result of SCSA and SCSA-OPT. In all cases they both converged to the same result, and in every case the policy to which SCSA converged could not be improved any further by PLI. This suggest that the convexity of \( V_k(s) \) may result in desirable characteristics for \( g_{\Psi, t}(\overrightarrow{T}) \). Based on this observation, we conjecture that \( g_{\Psi, t}(\overrightarrow{T}) \) has no local minimum.

In the next section, the convergence speed and the quality of the policies generated by SCSA will be studied through numerical experiments.

### 6.8 Numerical Results

We conduct simulation experiments to study the performance of structured admission control. For each set of system parameters, simulation has two stages. First, SCSA, implemented in Matlab, is used to obtain an optimal policy. Then, this policy is fed to a discrete-event simulator, built in C++, to find the resultant QoS performance. In the discrete event simulator, an actual model consisting of multiple APs (and not the approximate overflow model) is used. For each data point in the following charts, a total of \( 10^7 \) seconds are simulated. This is long enough to obtain stable performance measurements. All mobility, traffic, and system configuration parameters take their default values shown in Table 6.2 unless otherwise stated. We consider the underlay mesh topology given in Fig. 6.1, consisting of four APs. We assume
\[
\phi(n_1, n_2) = \frac{1}{6}
\]
for every pair of adjacent APs.

### 6.8.1 Optimality of Structured Admission Control

We have conducted an extensive number of experiments to compare the result of SCSA with SCSA-OPT and Value Iteration (VI) [45] under the approximate model of Section 6.5. All threshold levels obtained by using SCSA, SCSA-OPT and VI showed complete matching. An intuitive explanation for the exact matching is that due to the smooth and monotonic nature of the optimal threshold curves proved in Theorems 6.6.1–6.6.3, SCSA gradually converges to the optimal policy, rather than sub-optimal solutions.

The above observations suggest that, for all practical parameter sets, the proposed structured algorithm can be considered as a highly efficient yet reliable method to perform CAC in HWNs. As an example, for the system model in Fig. 6.1, with four mesh APs \( M = 4 \), the computed threshold policies for different arrival rates are given in Table 6.3. Another example

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_c )</td>
<td>50 calls</td>
<td>( \mu_c )</td>
<td>0.01</td>
</tr>
<tr>
<td>( C_w )</td>
<td>20 calls</td>
<td>( \mu_w )</td>
<td>0.01</td>
</tr>
<tr>
<td>( \lambda_{nc} )</td>
<td>0.4 calls/sec</td>
<td>( C_{BNC} )</td>
<td>5</td>
</tr>
<tr>
<td>( \lambda_{nw} )</td>
<td>0.2 calls/sec</td>
<td>( C_{BNO} )</td>
<td>10</td>
</tr>
<tr>
<td>( \eta_{hcc} )</td>
<td>( 5 \times 10^{-3} )</td>
<td>( C_{DCC} )</td>
<td>20</td>
</tr>
<tr>
<td>( \eta_{h} )</td>
<td>0.01</td>
<td>( C_{DHO} )</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 6.2: Simulation parameters.
<table>
<thead>
<tr>
<th>$l=1$</th>
<th>$n=0$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>36</td>
<td>25</td>
<td>11</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>36</td>
<td>25</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>50</td>
<td>36</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(a) $\lambda_{nc} = 0.8$ and $\lambda_{nw} = 0.6$

<table>
<thead>
<tr>
<th>$l=1$</th>
<th>$n=0$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38</td>
<td>38</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>38</td>
<td>38</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>50</td>
<td>38</td>
<td>38</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(b) $\lambda_{nc} = 0.6$ and $\lambda_{nw} = 0.4$

<table>
<thead>
<tr>
<th>$l=1$</th>
<th>$n=0$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>28</td>
<td>28</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>45</td>
<td>28</td>
<td>28</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>45</td>
<td>45</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(c) $\lambda_{nc} = 0.4$ and $\lambda_{nw} = 0.2$

<table>
<thead>
<tr>
<th>$l=1$</th>
<th>$n=0$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
<td>37</td>
<td>33</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>47</td>
<td>46</td>
<td>37</td>
<td>33</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>47</td>
<td>46</td>
<td>37</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(d) $\lambda_{nc} = 0.3$ and $\lambda_{nw} = 0.2$

<table>
<thead>
<tr>
<th>$l=1$</th>
<th>$n=0$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>49</td>
<td>43</td>
<td>42</td>
<td>21</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>49</td>
<td>49</td>
<td>43</td>
<td>42</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>49</td>
<td>49</td>
<td>43</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(e) $\lambda_{nc} = 0.2$ and $\lambda_{nw} = 0.1$

<table>
<thead>
<tr>
<th>$l=1$</th>
<th>$n=0$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(f) $\lambda_{nc} = 0.1$ and $\lambda_{nw} = 0.05$

Table 6.3: Threshold policies for $M = 4$ and different arrival rates.
### Table 6.4: Threshold policies for $M = 8$, $\eta = 5 \times 10^{-3}$ and different arrival rates.

<table>
<thead>
<tr>
<th>$P(l, n)$</th>
<th>n= 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l=1$</td>
<td>41</td>
<td>21</td>
<td>21</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=2$</td>
<td>41</td>
<td>41</td>
<td>21</td>
<td>21</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=3$</td>
<td>50</td>
<td>41</td>
<td>41</td>
<td>21</td>
<td>21</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=4$</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(a) $\lambda_{nc} = 0.8$ and $\lambda_{nw} = 0.6$

<table>
<thead>
<tr>
<th>$P(l, n)$</th>
<th>n= 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l=1$</td>
<td>41</td>
<td>35</td>
<td>24</td>
<td>14</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=2$</td>
<td>47</td>
<td>41</td>
<td>35</td>
<td>24</td>
<td>14</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=3$</td>
<td>48</td>
<td>47</td>
<td>41</td>
<td>35</td>
<td>24</td>
<td>14</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$l=4$</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(b) $\lambda_{nc} = 0.6$ and $\lambda_{nw} = 0.4$

<table>
<thead>
<tr>
<th>$P(l, n)$</th>
<th>n= 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l=1$</td>
<td>44</td>
<td>30</td>
<td>30</td>
<td>13</td>
<td>13</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=2$</td>
<td>44</td>
<td>44</td>
<td>30</td>
<td>30</td>
<td>13</td>
<td>13</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$l=3$</td>
<td>50</td>
<td>44</td>
<td>44</td>
<td>30</td>
<td>30</td>
<td>13</td>
<td>13</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$l=4$</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(c) $\lambda_{nc} = 0.4$ and $\lambda_{nw} = 0.2$

<table>
<thead>
<tr>
<th>$P(l, n)$</th>
<th>n= 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l=1$</td>
<td>45</td>
<td>33</td>
<td>33</td>
<td>15</td>
<td>15</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$l=2$</td>
<td>45</td>
<td>45</td>
<td>33</td>
<td>33</td>
<td>15</td>
<td>15</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$l=3$</td>
<td>50</td>
<td>45</td>
<td>45</td>
<td>33</td>
<td>33</td>
<td>15</td>
<td>15</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$l=4$</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

(d) $\lambda_{nc} = 0.3$ and $\lambda_{nw} = 0.2$
Figure 6.6: Mesh network with 8 APs.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>n= 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^n_v$</td>
<td>-</td>
<td>0.0084</td>
<td>0.0062</td>
<td>0.0040</td>
<td>0.0020</td>
</tr>
<tr>
<td>$\beta^n_v$</td>
<td>-</td>
<td>0.1398</td>
<td>0.2803</td>
<td>0.4217</td>
<td>0.5639</td>
</tr>
<tr>
<td>$\lambda^n_{nv}$</td>
<td>0</td>
<td>0.200</td>
<td>0.400</td>
<td>0.600</td>
<td>0.800</td>
</tr>
<tr>
<td>$\lambda^n_{hv}$</td>
<td>0.2883</td>
<td>0.3430</td>
<td>0.3960</td>
<td>0.4471</td>
<td>0.4966</td>
</tr>
</tbody>
</table>

Table 6.5: MMPP($Q_v, \Lambda_v$) parameters.

is given in Table 6.4 for $M = 8$ where eight APs are placed on a line as depicted in Fig. 6.6.

The parameters of the overflow MMPP obtained by PO-MMPP modeling are given in Table 6.5. It is notable that $\lambda^n_{nv}$ starts at zero and increases linearly as $n$, the number of blocking mesh nodes, increases. In contrast, $\lambda^n_{hv}$ is non-zero at $n = 0$. This is the rate of calls leaving mesh coverage area to the single coverage area as given in Equation 6.8, and is independent of $n$. $\lambda^n_{hv}$ has yet another component, a burtsy overflow stream that is triggered only when $n > 0$. 
### 6.8.2 Convergence Speed

The convergence speed, in CPU time, of SCSA is compared to SCSA-OPT and VI in Table 6.6 under the approximate model of Section 6.5. The size of the state space of an exhaustive search (ES) for the optimal policy, which is of order $O(2^{MCc})$, is given in the last column. Clearly, knowledge of the policy structure allows us to compute the optimal policy much more quickly than either VI or ES. It is notable that the times reported in Table 6.6 are MATLAB execution times which can be significantly enhanced in practical implementations.

The convergence times of SCSA and SCSA-OPT are very close in all cases. This is due to the fact that the policy to which SCSA converged could not be improved any further by PLI step. The slight difference in the convergence speed for $M = 16$ is due to the time it takes to evaluate the condition in step 10 of Algorithm 6.

### 6.8.3 SCSA vs. Complete Sharing and Guard-Channel Policies

In this section, we compare the performance of SCSA against two widely used admission control methods called Complete Sharing (CS) policy and Guard-Channel (GH) policies. CS policy refers to the admission policy in which $\Psi(s) = 1$, i.e., the admission decision in every

<table>
<thead>
<tr>
<th>$M$</th>
<th>SCSA</th>
<th>SCSA-OPT</th>
<th>VI</th>
<th>ES</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>12</td>
<td>12</td>
<td>483</td>
<td>$2^{200}$</td>
</tr>
<tr>
<td>8</td>
<td>59</td>
<td>59</td>
<td>5685</td>
<td>$2^{400}$</td>
</tr>
<tr>
<td>16</td>
<td>307</td>
<td>309</td>
<td>$\gg5685$</td>
<td>$2^{800}$</td>
</tr>
</tbody>
</table>

Table 6.6: SCSA Convergence speed, compared with SCSA-OPT and VI.
state is to admit calls, regardless of the system state as long as it remains within system capacity boundaries. GH policy, on the other hand, is more restrictive scheme in which a threshold on the number of active calls is used to make admission decisions. To compute the parameters of the GH policy we use the fact that the optimal threshold for calls of class $l$, if we have no observation of $n$, is the expected value of $\leftarrow T(l, n)$ with regard to $n$, i.e., $\leftarrow T(l) = E_n[\leftarrow T(l, n)]$. In other words, we assume the thresholds of an optimal GH policy are the expected values of thresholds given by SCSA, with expectation taken over the state of underlay mesh network.

In Figures 6.7a and 6.7b, $g_\Psi$, performance of SCSA, CS, and GH policies are compared for $\eta_h = 5 \times 10^{-3}$ and two different loading conditions over a range of $\lambda_{nc}$. In Fig. 6.7a, it is assumed that $\lambda_{nw} = \frac{1}{2} \lambda_{ne}$ and in Fig. 6.7b, we have $\lambda_{nw} = \frac{1}{3} \lambda_{ne}$. The above parameters translate to a system operating condition with acceptable QoS levels. For the case of $\lambda_{nw} = \frac{1}{2} \lambda_{ne}$, Fig. 6.8 illustrates blocking and dropping probabilities for call classes listed in Table 6.1.

These figures show that, even though SCSA operates over a reduced state space, it can significantly improve system performance. While it is observed that in all cases using SCSA results in a lower total cost, its potential savings become more significant when system load increases.

6.8.4 Partially Observable vs. Poisson Overflow Modeling

Figures 6.7a and 6.7b reflect the relative gain of using a partially observable model over a simple Poisson overflow model. In both figures, the curve for SCSA represents the performance of a controlled PO-MMPP/M/C/C queuing system, while the curve for GH policy represents the performance of a controlled M/M/C/C system, where in the latter, the overflow is modeled using a simple Poisson process. It can be seen that even limited knowledge of mesh network status can be used to significantly improve the system performance.
Figure 6.7: SCSA performance.
6.8.5 QoS Performance

In this section, we extend the experiment in Section 6.8.3 further by investigating the effect of increasing arrival rates on call dropping and blocking probabilities. While the system is exposed to an increasing arrival rate, $\lambda_{nc}$, we maintain $\lambda_{nw} = \frac{1}{2} \lambda_{nc}$. The results are given in Figures 6.9 and 6.10.

Our first observation is that higher arrival rates, clearly, result in higher dropping and blocking probabilities. $P_{DHO}$, dropping probability of handover overflows and $P_{BNO}$, blocking probability of new overflows, are both increasing with increasing $\lambda_{nc}$. As expected for the CS policy, these probabilities are equal. Second observation is that there is always a trade-off. Significant reductions in $P_{DHO}$ come at the expense of increased $P_{BNC}$ and $P_{BNO}$ as compared to CS policy. However, an efficient control algorithm has to strike a fine balance between these conflicting goals such that the desired QoS levels are achieved with minimal impacts on other performance metrics. Results presented in Section 6.8.3 confirm that SCSA, while improving
certain QoS metrics, yields better aggregate performance compared to CS or GH policies.

Another notable observation is that while SCSA provides a lower $P_{DHO}$ as compared to GH, it also manages to provide a lower $P_{BNO}$. Moreover, the reduction in $P_{DHO}$ for SCSA over GH is not as high as the relative reduction in $P_{BNO}$. This implies that SCSA is efficient in accommodating lower-priority calls in the presence of higher-priority classes.

In the next experiment, we study the effect of changing rejection cost ratios. As given in Table 6.1, we associate rejection costs to call classes. SCSA finds the policy that minimizes the system running cost, which is the total sum of rejection costs over time. Based on the ratio between these costs, call classes will receive preferential treatment. For example, as we increase $C_{DHO}$, making it more expensive to reject calls of this class, we anticipate to see $P_{DHO}$ drop. However, this will come at the expense of increasing blocking rates for other call classes.

For the arrival rate, we choose $\lambda_{nc} = 0.5$ and $\lambda_{nw} = 0.25$ and assume $C_{BNC} = C_{BNO} = C_{DCC} = 5$, while $C_{DHO}$ takes on different values. Figure 6.11 shows the result of this experiment. It can be observed that a reduction in $P_{DHO}$ is achieved with minimal increase in the blocking rate of other call classes. This indicates that SCSA is efficient in prioritizing call classes for a variety of cost ratios. Another notable observation is that we can use SCSA to reduce $P_{DHO}$ down to a desired level, by increasing its rejection cost.

### 6.8.6 Effect of User Mobility Rate

In this section, we study the performance of SCSA versus CS and GH for different mobility rates. Simulation parameters assume their default values as given in Table 6.2. The system is exposed to different mobility rates $\eta_h$ to study its response. The results are given in Figure 6.16.

In Figures 6.12a and 6.12b, the performance $g_\Psi$ of SCSA, CS, and GH policies are compared for two different loading conditions over a range of $\eta_h$. In Fig. 6.12a, we have $\lambda_{nc} = 0.3$ and $\lambda_{nw} = 0.15$ and in Fig. 6.12b, we have $\lambda_{nc} = 0.3$ and $\lambda_{nw} = 0.1$. Here, $\eta_h$ is the intensity of mobility as it measures the rate of leaving an underlay AP due to handover. We have chosen
Figure 6.9: Comparison of SCSA, CS and GH QoS performance.
Figure 6.10: Comparison of SCSA, CS and GH QoS performance.
Figure 6.11: Dropping and blocking probabilities for variable relative cost.

a range of $10^{-3} \ldots 10^{-2}$ for $\eta_h$ which corresponds to a range of 100 seconds to 1000 seconds for mobile station’s average residence time.

Clearly, increasing the mobility rate results in more frequent handovers which are resource consuming, and therefore it results in a higher system running cost. It can be observed that SCSA outperforms both CS and GH policies for a variety of different mobility rates. It is notable that as the system load increases, the gap between GH and SCSA performance widens. This reinforces our view that SCSA relative gains are more pronounced in higher loads.

6.8.7 Exact vs. Approximate Model

In Section 6.5, we employed an approximate model of overflow traffic. If we were to use an exact model, the size of state space will increase from $C^c M$ to $C^c C^w^M$. For $M = 4$ and $C^w = 20$, this translates to an increase of 40,000 times. From Table 6.7.1, we know that it takes 483 seconds to perform VI on a space of size $C^c M = 200$. Performing VI on an exact
Figure 6.12: Effect of user mobility rate on SCSA performance.
system model with the state space size of $C_c C_M = 8 \times 10^6$ will take at least $2 \times 10^7$ seconds or roughly 231 days. This is all assuming that the convergence time of VI will be linear in the size of state space, while it is well know, and can be observed from Table 6.7.1, that the time increases hyper-linearly as the state space grows.

This excessive convergence time and the amount of memory needed to store intermediate results and transition probability matrices make it impossible to use the exact model in any practical settings.

### 6.9 Discussion

In the analysis of overflow traffic in queuing networks, it is common to model each queue as an IPP, and to model the aggregate overflow as an MMPP source. As explained in Section 2.4, it is generally assumed that underlying IPPs are statistically independent. This is required to keep the mathematical framework tractable. While this assumption might be correct in some scenarios, clearly, there are cases in which underlying IPPs are correlated. In such scenarios, using MMPP as an approximate model for the aggregate overflow could introduce modeling error. As such, it is necessary to investigate the accuracy of this modeling technique.
In this section, we construct two queuing networks where one is an approximate version of the other one. The first network, which represents the exact model, is depicted in Fig. 6.13a, and consists of an overlay queue, receiving overflow traffic from four underlying M/M/C/C queues. The capacity of underlay APs is assumed to be $C_w = 20$ and the capacity of overlay is assumed to be $C_c = 50$. We also assume that call durations are exponentially distributed at the overlay and underlay with averages of $\mu_c$ and $\mu_w$, respectively. We use a memoryless Markovian mobility model such that mobile users leave an AP’s coverage area within an exponentially distributed time with the rate of $\eta_h$. To simulate mobile user movements, a discrete event simulator is used, that generates events at exponentially-distributed times to schedule the transition from one AP to another. The second network, shown in Fig. 6.13b, is an MMPP approximate of the first network, in which we replace the four underlying queues with one equivalent MMPP overflow stream. Clearly, in the first network, underlying queues are not independent because a call dropped in one queue (AP, here) will never get to visit another queue.

The performance metric we are interested in is the blocking probability of overflow traffic at the overlay queue. To find that, we use a discrete event simulator. Further details on this discrete event simulator are given in Section 3.2. For the first system, we use an exact model

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_c$</td>
<td>50 calls</td>
<td>$\mu_c$</td>
<td>0.01</td>
</tr>
<tr>
<td>$C_w$</td>
<td>20 calls</td>
<td>$\mu_w$</td>
<td>0.01</td>
</tr>
<tr>
<td>$\lambda_{nc}$</td>
<td>0 calls/sec</td>
<td>$\eta_h$</td>
<td>0.005</td>
</tr>
<tr>
<td>$\lambda_{nw}$</td>
<td>0.2 calls/sec</td>
<td>$T$</td>
<td>$10^7$ sec</td>
</tr>
</tbody>
</table>

Table 6.7: Simulation parameters.
that accounts for the correlation between overflow streams by simulating the exact behavior of calls moving from one queue to another. All parameters take their default values as given in Table 6.7. We assume that $\lambda_{nc} = 0$, i.e., there is no exogenous traffic coming to overlay.

For the second system, we need to find the parameters of the equivalent MMPP. The derivation is similar to what is given in Equations 6.9-6.13. In this case, the overlay queue can be characterized as an MMPP/M/C/C queue. The parameters of the equivalent MMPP are given in Table 6.8 for $\lambda_{nw} = 0.2$.

To compare the blocking performance of these two models, we measure the blocking probability of overflow calls over a range of new call arrival rates to underlay queues, $\lambda_{nw}$. The result is given in Fig 6.14. Also, a semi-log plot of the blocking probabilities is given in Fig 6.15. It can be observed that the error introduced by using the MMPP approximate model is generally minimal, and for the blocking probabilities of practical relevance, i.e., $P_b \leq 0.01$, the error is negligible. It can also be seen that MMPP model overestimates blocking probabilities by a small margin. This overestimation, if MMPP is used in the design of admission control algorithms, results in a slight over-reservation of resources leading to better QoS for overflow calls at the expense of under-utilizing system resources. From Fig 6.15, we see that as the blocking probability increases the relative gap between the two models shrinks. It is notable that this

<table>
<thead>
<tr>
<th>Parameter</th>
<th>n= 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^n_v$</td>
<td>-</td>
<td>0.0734</td>
<td>0.0470</td>
<td>0.0242</td>
<td>0.0103</td>
</tr>
<tr>
<td>$\beta^n_v$</td>
<td>-</td>
<td>0.1392</td>
<td>0.2764</td>
<td>0.4087</td>
<td>0.5389</td>
</tr>
<tr>
<td>$\lambda^n_{nv}$</td>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>$\lambda^n_{hv}$</td>
<td>0</td>
<td>0.19494</td>
<td>0.37285</td>
<td>0.5091</td>
<td>0.62832</td>
</tr>
</tbody>
</table>

Table 6.8: Equivalent MMPP parameters for $\lambda_{nw} = 0.2$. 
modeling error is highly dependent on the specifics of network topology and users’ mobility pattern.

Furthermore, for different arrivals rates, $\lambda_{nw}$, the mean and the variance of inter-overflow intervals are given in Figures 6.16a and 6.16b. It can be observed that the state aggregation technique performed in Section 6.5.2 maintains the mean of the arrival rate but results in a higher variance. This increased variance, in turn, contributes to higher estimates of blocking probability for the approximate MMPP model.

6.10 Chapter Summary

In this chapter, we characterize the structure of optimal admission control policies for Heterogeneous Wireless Networks (HWN), consisting of an integration of wireless mesh networks with an overlaying cellular infrastructure. A new bursty PO-MMPP traffic model is developed. This model captures the burstiness of the overflow traffic under the imperfect observability of the mesh network states.

For this system, structural optimal control results are presented. These results are used to design an efficient computational algorithm to determine the call admission policy. The perfor-
Figure 6.15: Semi-log plot of overflow blocking rates for the exact and approximate model.

Figure 6.16: Mean and variance of inter-overflow intervals.
mance of this algorithm is compared against the well-known Value Iteration technique, as well as the Complete Sharing and Guard-Channel policies. Discrete-event call-level simulations confirm that the obtained policy is effective in maintaining the desired QoS performance.
6.11 Nomenclature
### Symbol Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_c$</td>
<td>Capacity of overlay cellular BS (in BBU)</td>
</tr>
<tr>
<td>$C_w$</td>
<td>Capacity of underlay mesh AP (in BBU)</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of APs in underlay</td>
</tr>
<tr>
<td>$\mu_c$</td>
<td>Call holding time rate at overlay</td>
</tr>
<tr>
<td>$\mu_w$</td>
<td>Call holding time rate at underlay</td>
</tr>
<tr>
<td>$\lambda_{nc}$</td>
<td>Arrival rate of new calls to overlay</td>
</tr>
<tr>
<td>$\lambda_{i,m}^m$</td>
<td>Handoff arrival rate from neighboring overlay cells</td>
</tr>
<tr>
<td>$\lambda_{i,u}^i$</td>
<td>Arrival rate of new calls to underlay AP $i$</td>
</tr>
<tr>
<td>$\lambda_{ho}^i$</td>
<td>Rate of handoff overflows from AP $i$</td>
</tr>
<tr>
<td>$\lambda_{no}^i$</td>
<td>Rate of new call overflows from AP $i$</td>
</tr>
<tr>
<td>$\lambda_{hw}^i$</td>
<td>Rate of calls leaving AP $i$</td>
</tr>
<tr>
<td>$\eta_{hwc}^i$</td>
<td>Handoff rate from AP $i$ to overlay</td>
</tr>
<tr>
<td>$\eta_{hw}^{i,j}$</td>
<td>Handoff rate from AP $i$ to AP $j$</td>
</tr>
<tr>
<td>$\eta_{hcc}$</td>
<td>Handoff rate to neighboring overlay cells</td>
</tr>
<tr>
<td>$\eta_h^i$</td>
<td>Handoff rate out of underlay AP $i$</td>
</tr>
<tr>
<td>$P_D^{(i)}$</td>
<td>Handoff calls dropping probability at AP $i$</td>
</tr>
<tr>
<td>$P_B^{(i)}$</td>
<td>New calls blocking probability at AP $i$</td>
</tr>
<tr>
<td>$\phi(i, j)$</td>
<td>Fraction of calls leaving AP $i$, going to AP $j$</td>
</tr>
</tbody>
</table>

Table 6.9: Symbols used and their description.
Chapter 7

Concluding Remarks

7.1 Conclusion

It is anticipated that, in the near future, Heterogeneous Wireless Networks (HWN) will be widely deployed, and mobile handsets will be capable of using a diverse range of communications technologies to remain "always best connected" [90]. Due to their inherent heterogeneity, the efficient deployment of these networks and the dynamic management of their communication resources will require a new generation of algorithms and schemes. This thesis addressed the challenge.

We introduced a decision theoretic framework that can be used to analyze and evaluate call admission control algorithms. This framework was further used to study the behavior of network operating cost metrics. We rigorously demonstrated that these metrics have certain characteristics that can be used to design very efficient control algorithms. We proposed several algorithms, as natural extensions to the conventional value iteration and policy iteration algorithms. These algorithms, while superior in their time-complexity, almost always converge to the optimal solution.

Our analytical results confirm not only the usefulness but also the optimality of some of the most practical assumptions used by network engineers. Engineers commonly use threshold
policies in designing admission control algorithms. A threshold policy relies on a cut-off line to make admission decisions. We rigorously proved that a multi-dimensional threshold policy can be used to achieve optimal performance in HWNs.

As a part of this work, we developed a discrete-event network simulator used to study the performance of difference control algorithms. Our numerical results showed that the proposed algorithms are efficient in terms of time-complexity and in achieving the optimal performance by significantly reducing the probability of dropped and blocked calls.

7.2 Future Work

As a final thought, we would like to highlight a few research directions for future studies. Intelligent application of decision theory and dynamic programming has a lot of potential to enhance our understanding of how computer networks should be controlled. The field of wireless networking is rather young at this stage. Most theories and applications are very ad-hoc in nature. For this field to reach its full potential we will need to have better guidelines, better insight on how the resources should be managed in these networks, and in essence strong and generic rules of thumb that would help engineers devise practical decisions efficiently. The following topics and directions, while not covered in this thesis, were explored to some extent during the course of this research study. We believe they have great potential, and as such we like to recount them here to benefit readers and future researchers.

- Phase-Type Distributions

Phase-type (PH) modeling and analysis will enable us to study a wider range of systems. Here the idea is that we can represent any service time or inter-arrival time distribution (by matching moments) as the absorption time of a Markov chain. Although using PH models adds to the numerical complexity of MDP problems, it will allow us to study optimal control algorithms for the systems in which arrival, service, or mobility processes are not memoryless [91–94].
• Product-Form and Insensitive Queuing Models

We restricted most of our analysis to memoryless process, i.e., Poisson arrival, or Markov-modulated Poisson arrival, exponential service times and memoryless mobility patterns. While for some systems, these might not be very accurate, there are certain classes of networking and wireless systems whose performance is insensitive to the exact distribution of involved processes and its higher moments beyond the first moment (average waiting time, as an example). These systems, being insensitive to the exact model used in the analysis can be studied using memoryless process that are mathematically tractable, knowing that the analysis will be correct for more complex processes. Here, the challenge is to show that performance metrics of a system are “insensitive”. A necessary and sufficient condition for insensitivity is if local balance equations hold for a system transition equations [95–98].

In addition, for an insensitive system, we can find the optimal policy assuming exponential inter-event times, and have the guarantee that even if the distribution started to deviate from exponential, the policy will perform well.

• Partially Observable MDPs

As the size of HWNs grows, it becomes impractical to assume that we can make control decisions knowing the system’s global state. Essentially, the communications overhead will be excessive, and even if we could collect all that data, the size of state space will be exceedingly large. An alternative is to consider partially observable MDPs. These models can deal with situations in which we do not know the exact system state with certainty, and instead, we have partial knowledge (such as a time-variant probability distribution) of what the state might be. This is in contrast to earlier MDP models where we assumed that we can observe the system state. As an example, while we assumed that we can observe the number of ongoing calls in a WLAN or cellular network, in a large mesh network it might not be practical to assume that we can know the whole system state. In such complex HWNs, partially observable MDPs can be utilized to overcome
scalability issues [99–101].
Appendix A

Integration with Single Access Point

A.1 Proof of Lemma 5.3.1

Proof. \( V_{k+1}(i, j) \) as shown in (5.2) consists of 9 terms, \( T_1 \ldots T_9 \). Here for clarity, we factor out the common \( 1/v_{\text{max}} \) term and label the costs by their class number. We define

\[
D_m(i, j) = T_m(i, j) - T_m(i - 1, j) \quad \text{(A.1)}
\]

\[
V_{k+1}(i, j) = \frac{1}{v_{\text{max}}} \sum_{m=1}^{9} T_m(i, j) \quad \text{(A.2)}
\]

\[
\Delta^i V_{k+1}(i, j) = \frac{1}{v_{\text{max}}} \sum_{m=1}^{9} D_m(i, j). \quad \text{(A.3)}
\]

We use induction on \( k \) to prove \( V_{k+1}(i, j) \) is convex and monotonically non-decreasing in \( i \) for every fixed \( j \) (proof for convexity in \( j \) is similar). This is equal to showing \( \Delta^i V_{k+1}(i + 1, j) \geq \Delta^i V_{k+1}(i, j) \) and \( \Delta^i V_{k+1}(i, j) \geq 0 \). The induction hypothesis is the following: \( V_k(i, j) \) is (I) monotonically non-decreasing in \( i \) and \( j \) and (II) convex in \( i \) (or \( j \)) for every fixed \( j \) (or \( i \)).

Proof of (I): We show \( \Delta^i V_{k+1}(i, j) \geq 0 \) by evaluating each \( D_m \) term individually. The basis step is trivial since \( \forall (i, j) \leq (C_c, C_w) : V_0(i, j) = 0 \) and \( \forall j : V_0(C + 1, j) = \infty \). Consider
\(D_1(i, j)\), when \(\lambda_c\) is factored out. Let

\[
D'_1(i, j) = \frac{[T_1(i, j) - T_1(i - 1, j)]}{\lambda_c}
\]

\[
= \min(V_k(i, j) + C_1, V_k(i + 1, j))
- \min(V_k(i - 1, j) + C_1, V_k(i, j))
\]

\[
= \min(C_1, V_k(i + 1, j) - V_k(i, j)) + V_k(i, j)
\]

\[
\geq 0 \quad \text{by hypo. (I) for } V_k
\]

\[
- \min(C_1, V_k(i, j) - V_k(i - 1, j)) - V_k(i - 1, j)
\]

\[
\geq 0 \quad \text{by hypo. (I) for } V_k.
\]

\[(A.4)\]

The same method can be applied directly to \(T_{\{2,3,5,7\}}\). However, the rest of the terms have to be considered altogether. We first extend \(D_9(i, j)\) as follows,

\[
D_9(i, j) = T_9(i, j) - T_9(i - 1, j)
\]

\[
= (v_{\text{max}} - \nu_{\text{out}}(i, j))V_k(i, j)
- (v_{\text{max}} - \nu_{\text{out}}(i - 1, j))V_k(i - 1, j)
\]

\[
= (C_c - i)\eta_{\text{hec}} V_k(i, j) - (C_c - i + 1)\eta_{\text{hec}} V_k(i - 1, j)
+ (C_c - i)\eta_{\text{hec}} V_k(i, j) - (C_c - i + 1)\eta_{\text{hec}} V_k(i - 1, j)
\]

\[
+ (C_w - j)\eta_{\text{hwc}} V_k(i, j) - (C_w - j)\eta_{\text{hwc}} V_k(i - 1, j)
+ (C_c - i)\mu_c V_k(i, j) - (C_c - i + 1)\mu_c V_k(i - 1, j)
\]

\[
+ (C_w - j)\mu_w V_k(i, j) - (C_w - j)\mu_w V_k(i - 1, j).
\]

Although it is straightforward to show that the 3rd and 5th terms above are non-negative, for other terms it is not trivial. We show that every of these terms in \(D_9(i, j)\) if combined with other terms in \(\Delta'V_k(i, j)\) can be proved to be non-negative. As an example, consider the first
term which we can rewrite as

\[
D_9^{\text{hcc}} = (C_c - i)\eta_{\text{hcc}} V_k(i, j) - (C_c - i + 1)\eta_{\text{hcc}} V_k(i - 1, j)
\]

\[
= (C_c - i)\eta_{\text{hcc}} \underbrace{\{V_k(i, j) - V_k(i - 1, j)\}}_{\Delta V_k(i, j) \geq 0} - \eta_{\text{hcc}} V_k(i - 1, j).
\]

\[(A.6)\]

Equation A.6 contains \(\eta_{\text{hcc}}\), and from (5.2) we see that term \(T_8\) has \(\eta_{\text{hcc}}\) as well, so we will use \(D_8(i, j)\). We consider the sum of them:

\[
D_9^{\text{hcc}} + D_8(i, j) =
\]

\[
(C_c - i)\eta_{\text{hcc}} \{V_k(i, j) - V_k(i - 1, j)\} - \eta_{\text{hcc}} V_k(i - 1, j) + i\eta_{\text{hcc}} V_k(i - 1, j) - (i - 1)\eta_{\text{hcc}} V_k(i - 2, j)
\]

\[
= (C_c - i)\eta_{\text{hcc}} \{V_k(i, j) - V_k(i - 1, j)\} + (i - 1)\eta_{\text{hcc}} \Delta^i V_k(i - 1, j)
\]

\[
\geq 0.
\]

\[(A.7)\]

The other two terms in \(D_9(i, j)\) can be matched with \(T_4\) and \(T_6\) similarly. Hence, \(\Delta^i V_{k+1}(i, j) = \sum_{m=1}^{9} D_m \geq 0.\)

**Proof of (II):** We need to show \(\Delta^i V_{k+1}(i+1, j) \geq \Delta^i V_{k+1}(i, j)\). This is equal to \(\sum_{m=1}^{9} [D_m(i+1, j) - D_m(i, j)] \geq 0.\) Similar to the last part, the basis step is trivial since \(\forall (i, j) \leq (C_c, C_w) : V_0(i, j) = 0\) and \(\forall j : V_0(C + 1, j) = \infty\). Let us define \(Y_m(i, j) = D_m(i, j) - D_m(i - 1, j)\). Again, we prove terms are non-negative either individually or when combined with other terms.
Let us start with $Y_1$, having factored out $\lambda_c$:

$$Y'_1(i + 1, j) = \frac{[D_1(i + 1, j) - D_1(i, j)]}{\lambda_c}$$

$$= \min(C_1, V_k(i + 2, j) - V_k(i + 1, j)) + V_k(i + 1, j)$$

$$- \min(C_1, V_k(i + 1, j) - V_k(i, j)) - V_k(i, j)$$

$$- \min(C_1, V_k(i + 1, j) - V_k(i, j)) - V_k(i, j)$$

$$+ \min(C_1, V_k(i, j) - V_k(i - 1, j)) + V_k(i - 1, j)$$

$$= \frac{\min(C_1, \Delta^i V_k(i + 2, j)) - \min(C_1, \Delta^i V_k(i + 1, j))}{\geq 0 \text{ by hypo. (II) for } V_k}$$

$$+ \Delta^i V_k(i + 1, j) - \Delta^i V_k(i, j)$$

$$- [\min(C_1, \Delta^i V_k(i + 1, j)) - \min(C_1, \Delta^i V_k(i, j))]$$

$$\geq \Delta^i V_k(i + 1, j) - \Delta^i V_k(i, j)$$

$$- [\min(C_1, \Delta^i V_k(i + 1, j)) - \min(C_1, \Delta^i V_k(i, j))]$$

$$\geq 0 \text{ (by hypo. (II) for } V_k). \quad (A.8)$$

Again, the proof for $Y_{\{2,3,5,7\}}$ is the same as for $Y_1$. For other terms, we have to take an approach similar to the last part. Let us extend $Y_9$, focusing on the terms containing $\eta_{hec}$:

$$Y_9(i + 1, j) = D_9(i + 1, j) - D_9(i, j)$$

$$= (v_{\max} - v_{out}(i + 1, j)) V_k(i + 1, j)$$

$$- (v_{\max} - v_{out}(i, j)) V_k(i, j)$$

$$- \{(v_{\max} - v_{out}(i, j)) V_k(i, j)$$

$$- (v_{\max} - v_{out}(i - 1, j)) V_k(i - 1, j)\}$$

$$= (C_c - i - 1) \eta_{hec} V_k(i + 1, j) - (C_c - i) \eta_{hec} V_k(i, j)$$

$$- (C_c - i) \eta_{hec} V_k(i, j) + (C_c - i + 1) \eta_{hec} V_k(i - 1, j)$$

$$+ \ldots. \quad (A.9)$$
Separating the terms with $\eta_{hcc}$, we obtain

$$Y_9^{\eta_{hcc}}(i + 1, j) =$$

$$(C_c - i - 1)\eta_{hcc}V_k(i + 1, j) - (C_c - i)\eta_{hcc}V_k(i, j)$$

$$- (C_c - i)\eta_{hcc}V_k(i, j) + (C_c - i + 1)\eta_{hcc}V_k(i - 1, j)$$

$$= (C_c - i - 1)\eta_{hcc}\{V_k(i + 1, j) - V_k(i, j)\}$$

$$- (C_c - i - 1)\eta_{hcc}\{V_k(i, j) - V_k(i - 1, j)\}$$

$$- 2\eta_{hcc}V_k(i, j) + 2\eta_{hcc}V_k(i - 1, j).$$  \hspace{1cm} (A.10)

This term has to be evaluated along with $Y_8(i + 1, j)$ in order to show that the sum of both terms is non-negative. For $Y_8(i + 1, j)$ we have

$$Y_8(i + 1, j)$$

$$= (i + 1)\eta_{hcc}V_k(i, j) - i\eta_{hcc}V_k(i - 1, j)$$

$$- i\eta_{hcc}V_k(i - 1, j) + (i - 1)\eta_{hcc}V_k(i - 2, j)$$

$$= (i - 1)\eta_{hcc}\{V_k(i, j) - V_k(i - 1, j)\}$$

$$- (i - 1)\eta_{hcc}\{V_k(i - 1, j) - V_k(i - 2, j)\}$$

$$+ 2\eta_{hcc}V_k(i, j) - 2\eta_{hcc}V_k(i - 1, j),$$  \hspace{1cm} (A.11)
and the sum of these two terms is

\[ Y_9^{\eta_{hec}}(i+1,j) + Y_8(i+1,j) = \]
\[ = (C_c - i - 1)\eta_{hec} \left\{ V_k(i+1,j) - V_k(i,j) \right\} \]
\[ - (C_c - i - 1)\eta_{hec} \left\{ V_k(i,j) - V_k(i-1,j) \right\} \]
\[ + (i - 1)\eta_{hec} \left\{ V_k(i,j) - V_k(i-1,j) \right\} \]
\[ - (i - 1)\eta_{hec} \left\{ V_k(i-1,j) - V_k(i-2,j) \right\} \]
\[ = (C_c - i - 1)\eta_{hec} \left\{ \Delta^iV_k(i+1,j) - \Delta^iV_{k+1}(i,j) \right\} \]
\[ \geq 0 \quad \text{by hypo. (II) for } V_k \]
\[ + (i - 1)\eta_{hec} \left\{ \Delta^iV_k(i,j) - \Delta^iV_{k-1}(i,j) \right\} \]
\[ \geq 0 \quad \text{by hypo. (II) for } V_k \]
\[ \geq 0. \]  

(A.12)

It is similar to show non-negativity for the other terms of \( Y_m \). Since we have \( \forall m : Y_m \geq 0 \), the inequality \( \Delta^iV_{k+1}(i+1,j) \geq \Delta^iV_{k+1}(i,j) \) holds, and hence, the cost function is convex. \( \square \)
Appendix B

Integration with Wireless Mesh Network

B.1 Proof of Lemma 6.6.1

Proof. In this lemma we want to prove that $\lambda_{nv}^{(n)}$ and $\lambda_{hv}^{(n)}$ given in (6.17) are non-decreasing in $n$. We prove this for $\lambda_{nv}^{(n)}$. The proof for $\lambda_{hv}^{(n)}$ is similar. First, for the case when the system state is $\vec{u} = (x_1, \ldots, x_M)$, and $\vec{u} \in U(n)$, i.e, only $n$ IPPs are ON, let us denote by $m^{(h)}$, the index of the $h$th IPP that is ON. More formally, $m^{(h)} = \{ j : \sum_{i=1}^{j} x_i = h \}$.

Initially, we modeled every AP as an independent IPP in Equations (6.10-6.11). As the blocking/dropping state of an AP is independent of the state of other APs, we can directly find the stationary probability of finding an AP in the non-overflow state as

$$f^{(i)} \triangleq \Pr\{x_i = 1\} = \frac{\alpha_i}{\alpha_i + \beta_i}$$ \hspace{1cm} (B.1)

where $\alpha_i$ and $\beta_i$ are given in (6.11). To prove the monotonicity result, we have to show that $\lambda_{nv}^{(n)} \leq \lambda_{nv}^{(n+1)}$. For $\lambda_{nv}^{(n+1)}$ we have

$$\lambda_{nv}^{(n+1)} = \mathbb{E}_u[\lambda_{no}(\vec{u}) | \vec{u} \in U(n+1)] \geq \mathbb{E}_u[\lambda_{no}(\vec{u}) - \lambda_{no}^{\hat{r}} | \vec{u} \in U(n+1)].$$ \hspace{1cm} (B.2)

Recall that $\lambda_{no}(\vec{u}) = \sum_{i=1}^{M} x_i \lambda_{no}^i$. Random variable $\hat{r}$ is chosen from the indices of arrival rates included in $\lambda_{no}(\vec{u})$, i.e., $\hat{r} \in \{ m^{(1)}, \ldots, m^{(n+1)} \}$. Note that as the expectations are calculated
over \( \bar{u} \), the value of the second expectation is a random function of \( \hat{r} \). This term corresponds to the expected overflow rate of new calls if AP \( \hat{r} \) was artificially removed from the set of \( n + 1 \) APs generating overflow. We can choose any arbitrary distribution for \( \hat{r} \). However, one selection that has practical relevance is

\[
Pr\{\hat{r} = m^{(h)}\} = \frac{f(m^{(h)})}{\sum_{i=1}^{n+1} f(m^{(i)})}.
\]

This is the conditional probability of finding AP \( m^{(h)} \) at a non-overflow state if it has to be chosen among the set of \( n + 1 \) currently saturated APs. If we take the expectation over \( \hat{r} \) for both sides we arrive at

\[
\lambda_{nv}^{(n+1)} \geq Er_u[\lambda_{no}(\bar{u}) - \hat{\lambda}_{no} | \bar{u} \in U(n+1)].
\]

However, given the distribution in (B.3) for \( \hat{r} \), the above expectation is equal to the average overflow rate for the system if only \( n \) APs are found busy which is \( \lambda_{nv}^{(n)} \). Therefore, we have

\[
\lambda_{nv}^{(n+1)} \geq \lambda_{nv}^{(n)}.
\]

\[\square\]

**B.2 Proof of Lemma 6.6.2**

*Proof.* as shown in (6.25) consists of 8 terms, \( T_1(s) \ldots T_8(s) \). Here for clarity, we factor out the common \( 1/v_{\text{max}} \) term and label the costs by their class numbers. We define

\[
V_{k+1}(s) = \frac{1}{v_{\text{max}}} \sum_{m=1}^{8} T_m(s)
\]

\[
D_m(\psi_e s) = T_m(\psi_e s) - T_m(s)
\]

\[
\Delta V_{k+1}(\psi_e s) = \frac{1}{v_{\text{max}}} \sum_{m=1}^{8} D_m(\psi_e s).
\]

In what follows, we present the proof for Property B. The proof for Property A is simpler, and similar to the one given in [22]. The proof for Property C is similar to the proof given here for Property B. Note that Property A is needed to prove the more involving Properties B and C.
**Proof of Property B**: We use induction on \( k \) to prove that \( \Delta V_k(A^2s) \geq \Delta V_k(As) \). The basis step is trivial since \( \forall s \in S : V_0(s) = 0 \), and thus \( \Delta V_0(A^2s) = \Delta V_0(As) = 0 \). Assuming that the induction hypothesis holds for \( V_k(s) \), we want to show that \( V_{k+1}(A^2s) - V_{k+1}(As) \geq 0 \). We have

\[
\Delta V_{k+1}(As) = \frac{1}{v_{\text{max}}} \left\{ \lambda_{hec}^m \min[\Delta V_k(A^2s), C_{\text{DCC}}] - \lambda_{hec}^m \min[\Delta V_k(As), C_{\text{DCC}}] + \lambda_{he}^{(m)} \min[\Delta V_k(A^2s), C_{\text{DHO}}] - \lambda_{he}^{(m)} \min[\Delta V_k(As), C_{\text{DHO}}] \right\} \sum_{m=1}^{8} Y_m(A^2s) \]  

(B.8)

Let us define \( Y_m(s) \) such that

\[
\Delta V_{k+1}(A^2s) - \Delta V_{k+1}(As) = \frac{1}{v_{\text{max}}} \sum_{m=1}^{8} Y_m(A^2s)
\]

(B.9)

We continue by showing that either \( Y_m(s) \) is greater than zero for a given \( m \) or the sum of two of them is greater than zero. Let us consider the first term \( Y_1(A^2s) \). We have

\[
Y_1(A^2s) = \lambda_{hec}^m \{ \min[\Delta V_k(A^3s), C_{\text{DCC}}] + V_k(A^2s) - \min[\Delta V_k(A^2s), C_{\text{DCC}}] - V_k(As) - \min[\Delta V_k(A^2s), C_{\text{DCC}}] - V_k(As) + \min[\Delta V_k(As), C_{\text{DCC}}] + V_k(s) \} \]  

(B.10)

Here, the secondary terms of \( V_k(A^n s) \) are taken from \( Y_8(A^2s) \), i.e., the last term with \( v_{\text{max}} \).
coefficient. We can rewrite (B.10) as
\[
Y_1(A^2 s) = \lambda_{hcc}^{in} \left\{ \min\{\Delta V_k(A^3 s), C_{DCC}\} 
- \min[\Delta V_k(A^2 s), C_{DCC}] 
+ \frac{V_k(A^2 s) - V_k(A s) - \{V_k(A s) - V_k(s)\}}{\Delta V_k(A^2 s) - \Delta V_k(A s)} 
- \min[\Delta V_k(A^2 s), C_{DCC}] 
+ \min[\Delta V_k(A s), C_{DCC}] \right\} 
\]
(B.11)

The first and second terms are jointly greater than zero due to the induction hypothesis. The fourth and the fifth lines are the minimization of the variables in the third line. Therefore, \(Y_1(A^2 s) \geq 0\).

As another example, consider the seventh term \(Y_7(A^2 s)\). We have
\[
Y_7(A^2 s) = (\mu_c + \eta_{hcc}) \left[ (i + 2)\Delta V_k(\Delta A^2 s) 
- (i + 1)\Delta V_k(\Delta A s) 
- (i + 1)\Delta V_k(\Delta A s) + i\Delta V_k(D s) 
+ C_c\{\Delta V_k(A^2 s) - \Delta V_k(A s)\} \right] 
\]
(B.12)

Here, the last line is taken from the eighth term. It is easy to verify that \(\Delta V_k(\Delta A^2 s) = -\Delta V_k(A^2 s)\) and \(\Delta V_k(\Delta A s) = -\Delta V_k(A s)\). Also, from the induction hypothesis we have that \(\Delta V_k(A s) \geq \Delta V_k(D s)\). We can rewrite (B.12) as
\[
Y_7(A^2 s) = (\mu_c + \eta_{hcc}) \left[ 
+ i\{\Delta V_k(A s) - \Delta V_k(D s)\} \geq 0 
+ (C_c - i - 2)\{\Delta V_k(A^2 s) - \Delta V_k(A s)\} \geq 0 \right] 
\]
\[
\geq 0 \]
Bibliography


