A Decision-Theoretic Approach to Resource Allocation in Wireless Multimedia Networks

Zygmunt Haas, Joseph Y. Halpern, Li Li, Stephen B. Wicker
School of Electrical Engineering/Computer Science Dept.
Cornell University
Ithaca, New York 14853
haas@ee.cornell.edu, halpern@cs.cornell.edu,
lili@cs.cornell.edu, wicker@ee.cornell.edu

Abstract— **The allocation of scarce spectral resources to support as many user applications as possible while maintaining reasonable quality of service is a fundamental problem in wireless communication. We argue that the problem is best formulated in terms of decision theory. We propose a scheme that takes decision-theoretic concerns (like preferences) into account and discuss the difficulties and subtleties involved in applying standard techniques from the theory of Markov Decision Processes (MDPs) in constructing an algorithm that is decision-theoretically optimal. As an example of the proposed framework, we construct such an algorithm under some simplifying assumptions. Additionally, we present analysis and simulation results that show that our algorithm meets its design goals. Finally, we investigate how far from optimal one well-known heuristic is. The main contribution of our results is in providing insight and guidance for the design of near-optimal admission-control policies.**

I. Introduction

When a user makes a call in a wireless network, the system must decide whether to admit the call or block it.[1] What makes the decision difficult is that, to achieve some sense of optimality, one needs to consider the future status of the network resources and the pattern of the future arrival requests. For example, even if the network currently has sufficient resources to handle the call, admitting the call may result in other, already in-progress calls being dropped in the future. This problem is aggravated by the trend to decreasing cell sizes coupled with an increased demand for multimedia services. The reduction in cell size leads to increased spectrum-utilization efficiency, but also increases the number of handoffs experienced during a typical connection. This means that the call requests at one cell will be more affected by decisions made in nearby cells. Furthermore, as the number of cells increases, the amount of resources per cell decreases. Thus, the *trunking efficiency* is reduced, leading to more severe fluctuations in the quality of service (QoS). The provision of connection-level QoS raises a distinct yet dependent set of issues. Guarantees made by one cell place future burdens, due to handoffs, on the resources of other cells.

This particular problem is but one of many in the general area of admission-control policies. The crafting of admission policies is generally approached by focusing on a subset of the design issues, while ignoring others. For example, several schemes for wireless networks [3], [21], [18] have been proposed to provide connection-level QoS by basing the call-admission decision on a requirement that the handoff-dropping probability be kept below a certain level. Clearly, however, lowering the handoff-dropping probability will, in general, mean increasing the probability of calls being blocked. A major issue remains unaddressed: is it worth decreasing the handoff-dropping probability from, say, 2% to 1% if it means increasing the probability of blocking from 5% to 10%? In other words, what is an acceptable tradeoff between call blocking and call dropping probabilities?

Tolerance to blocking and dropping among users is also not addressed by a simple minimization of handoff-dropping probability. For example, compare a typical voice call with a file transfer. A user making a voice call may be mildly frustrated if she cannot connect to the server, but would be much more annoyed if the call were dropped halfway through the conversation. On the other hand, a user attempting to do a number of FTP file transfers may prefer to be connected immediately, even if this means a higher probability of being dropped, since this increases the probability of at least some rapid file transfers. Should the network treat voice and file transfer connections uniformly and provide a single dropping probability for both?

We believe that sensible admission decisions should take both utilities and probabilities into account, and are best formulated in decision-theoretic terms. In this paper, we

[1] In this paper, we use the term "user" to refer to the actual user or the application running on the user's mobile device; we use the generic term "call" to indicate an attempt to initiate a session, which could be of any medium type or types (e.g., voice, data, video, hybrid); we use the term "block" to mean rejecting new calls, the term "drop" to mean rejecting calls already in progress (e.g., during a handoff process), and the term "reject" to mean either block or drop.

describe a general decision-theoretic approach to call admission. Although we use a specific example to demonstrate our approach, our approach gives a framework that can accommodate various QoS requirements in the multimedia resource-allocation process. Our focus is on wireless networks, but the generalization to other contexts is immediate. We show that even a simplistic decision-theoretic approach to wireless network admission policies produces substantial performance improvement over several well-known heuristics.

The key ingredients in decision theory are *probability* and *utility*. Probability is a measure of the likelihood of an event, while the utility (which can be interpreted as the reward) measures the event's perceived importance. A decision-theoretic admission-control policy can take into account the relative utility of blocking and dropping, as well as other considerations, such as network utilization or service differentiation and delay requirements.

If we are to use a decision-theoretic approach, we must somehow determine the probabilities and utilities. We assume that the probabilities can be obtained by taking sufficient statistical measurements. Obtaining the utilities is somewhat more problematic. Utilities are subjective and different users may have different utilities for the same application. Moreover, there is the problem of what is known as *inter-subjective* utility: one user's utility of 10 may not be equivalent to another user's 10.

We sidestep these problems to some extent by assuming that we have only one user, which can be thought of as the system provider. Of course, it is in the system provider's interest to keep users as happy as possible. So we can assume that, to some extent, the system provider's utility is representative of a typical user's utilities. This is particularly true if, as we would expect, the system provider gives higher utilities to calls for which users are prepared to pay more.

To use our approach, the system provider must decide what the relative utilities of blocking vs. dropping ought to be. Presumably this will be done by weighing the profits lost from blocking a call, compared to the profits lost due to customers switching to a different provider as a result of having calls dropped too frequently (and possibly refunds due to dropped calls). In this paper, we assume that we are simply given the relevant utilities, while recognizing that obtaining them may be a nontrivial problem.

Once we have the relevant probabilities and utilities, we can employ Markov Decision Processes (MDPs) [24]. More specifically, we model the call-admission problem as an MDP, allowing us to use well-known techniques for finding the optimal call-admission policy. However, we must be careful in modeling the call-admission problem as an MDP. The standard techniques for finding the optimal policy in an MDP run in time polynomial in the size of the state space and the number of actions. A larger state space leads to a more accurate model of the system. But, if we are not careful, the state space can quickly become unmanageable. We discuss this issue in detail and show that under some simplifying and yet reasonably practical assumptions we can construct a manageable state space.

We are not the first to apply MDPs to the call-admission problem (see Section VI for other references). However, there are a number of subtleties that arise in using MDPs in this context that do not seem to have been considered earlier. For example, it is typically assumed that we are given the probabilities and utilities in an MDP, and that we must then construct an optimal admission policy. However, in a wireless network, the probabilities are themselves influenced by the admission policy we construct. For example, the probability that a call will be handed off from a neighboring cell depends on the probability that the neighboring cell will accept the call. This, in turn, depends on the admission policy, which was developed using assumptions about those probabilities. This is an issue that does not seem to have been considered in the literature on MDPs, perhaps because it was assumed that the probabilities of call blocking and dropping were so low that they could be ignored. Since we want our system to be able to deal with situations where the probability of blocking may be nonnegligible, we must address this problem as part of our solution. We do so by constructing a sequence of policies that converge to a limit. In the limit, we have a policy that is "locally" optimal with respect to a given probability measure and utility function, such that the probability measure is precisely that induced by the policy.

By finding a (locally) optimal policy using MDPs, we are able to make optimal tradeoffs between such concerns as providing the desired connection-level QoS and spectrum utilization. We show by an extensive series of simulations that our approach provides substantial performance improvement over one of the best heuristic approaches considered in the literature, given some standard assumptions about arrival times and holding times of calls. However, we do not view our major contribution as the performance improvements in this particular case. Rather, we consider our major contribution as the general decision-theoretic framework and the discussion regarding how to go about using it in practice. We believe that this framework provides us with a general methodology for evaluating admission-control policies and for characterizing optimal policies.

The rest of this paper is organized as follows. Section II discusses our assumptions about the underlying system and our use of QoS classes. Section III considers how the admission-control problem can be formulated as an MDP and the subtleties involved in doing so. Section IV shows how the optimal admission-control policy can be computed. Section V presents results from an extensive series of simulations that compares the performance of the decision-theoretic admission policy for the case of two QoS classes, one for data and one for video, with a well-known heuristic approach [4], [21]. Related work is described in Section VI, which also offers some concluding remarks.

II. Network Model and QoS Classes

We assume that the network consists of a cellular wireless portion and a wired backbone. The wireless portion consists of a set of cells, each cell contains a single base station (BS). All the BSs are connected to the same Mobile Switching Center (MSC). The MSC is responsible for switching calls between cells and for maintaining connections to the wired backbone. We focus solely on the wireless portion in this paper; an integrated admission-control scheme can be obtained by taking the wired portion into account.

We want to allow the network the possibility of treating different calls in different ways due to, for instance, user preferences and tolerances. We would certainly expect different traffic classes (e.g., voice, video, data) to be treated differently, but, as we observed earlier, we may also want to treat calls within the same traffic class in different ways.

Thus, we propose that each traffic class be partitioned into a number of QoS classes, so that the network provides K QoS classes altogether. We may further want to refine each QoS class into a number of layers. The more layers assigned to a traffic stream, the better QoS is provided. Partitions into layers give the system an option for further QoS differentiation beyond just the decision of whether or not to admit a call. This would, for instance, allow us to take advantage of recent advances in video coding [2], [19] by representing video (or audio) streams as multilayer scalable flows that can adapt dynamically to changing network and link conditions. Dealing with layers is a straightforward extension of the ideas presented here; for ease of exposition, we do not consider layers further in this paper.

We assume that each QoS class is associated with a set of numbers representing utilities: the utility (cost) per unit of time of handling a call in that service class, the negative utility (cost) of blocking a call, and the cost of dropping a call during a handoff. (The idea of associating a set of utilities with a QoS class was inspired by the QoS contract notion in [1].)

III. Formulating Admission Control as an MDP

Our goal is to construct an admission-control policy that decides, for each new call and handoff call, whether it should be admitted or rejected and, if it is admitted, at what QoS level. The possibility of queuing incoming calls for later admission is not considered in this paper. We want to find a policy that maximizes expected utility.

It is not sufficient to simply maximize the expected utility of current new call requests and current admitted calls. This does not take into account the future need for handoffs as active users move across cell boundaries. Such an approach would result in future expected utility being much lower than the current expected utility. Instead, we try to maximize the expected *total utility* (in a sense made precise below). This means we must consider utility streams as opposed to temporally isolated utility values.

We cannot just identify the total utility with the sum of utilities over time, since this is in general an infinite sum. There are two standard approaches in the literature to defining total utility [24]. The first is to discount the utility over time by some factor γ, with $0 < \gamma < 1$. That is, if u_k is the utility obtained at time k, we take the total utility to be $\sum_{k=1}^{\infty} \gamma^k u_k$. We then try to find the policy that maximizes the expected total utility. This definition is meant to capture the notion of present value in accounting: a dollar now is worth more than a dollar one time unit later. The smaller γ is, the more we weight the present. This is reasonable if the time units are reasonably large. For example, banks pay interest so that \$95 at year t can become \$100 at year $t+1$. This suggests that having \$100 at year $t+1$ is the same as having \$95 at year t; i.e., we can assign $\gamma = .95$ to capture this effect.

A second approach to compute the total utility seems more appropriate in our context, since we are dealing with small time intervals. The focus in this approach is the average utility per unit of time; then the total utility over M time units can just be taken to be M times the average utility. Thus, in this approach we try to maximize the expected value of $\lim_{M\to\infty}(\sum_{k=1}^{M} u_i)/M$.

Whichever approach in defining the total utility we use, we can formulate the problem of finding the policy which optimizes the expected total utility as a *Markov Decision Process* (MDP). Formally, an MDP is a tuple (S, A, T, R) consisting of a set S of states, a set A of actions, a *transition function* $T : S \times A \to \Pi(S)$, where $\Pi(S)$ is the set of probability distributions over S, and a *reward function* $R : S \times A \to \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers. Intuitively, $T(s, a)$ describes the probability of ending up in a state $s' \in S$ if we perform action a in state s, while $R(s, a)$

describes the immediate reward obtained if we perform action a in state s.

There are standard techniques for finding the optimal policy in MDPs, such as value iteration and policy iteration. These algorithms and a number of their variants have been studied in detail, and it is well understood what are the relative advantages of each algorithm (again, see [24]). We want to model the call-admission problem as an MDP, so as to be able to use all the power of MDPs. We now discuss how this can be done, considering the components of an MDP one by one. We pay particular attention to the problem of keeping the size of the state space manageable. Our discussion here is at a general level; we return to the specific issues raised here when we consider simulation results in Section V.

A. The State Space

We represent the state of the cell as the number of calls in progress in the cell. Decisions at a cell are made on the basis of this state. Clearly, the more information about the system we put into the state, the more accurate the model of the system is, and thus, the more accurate our estimation of the actual rewards received. For our algorithms to work, we also need to put enough information into the state to guarantee the *Markov property*; i.e., we want the probability of making a transition from one state to another to depend only on the current state, not on the previous history. On the other hand, the more details we put into the state, the more states there are, and the longer it will take to compute the optimal policy. If we put in too much details, the state space will be so large that we cannot compute the optimal policy in a reasonable length of time. Thus, there is a tradeoff that must be made when modeling the state; we may have to model only certain features, while ignoring others. A technique such as value iteration computes the optimal policy with respect to the state model. Of course, if our model of the state does not capture enough features of relevance to the problem, the optimal policy computed by the MDP may not be optimal in practice.

We briefly discuss some of these issues that arise in modeling the state here and the choices we made in our simulations.

- Should we include time in the state? Clearly, on one hand, traffic and mobility patterns depend on time of day (and day of the week and week of the year), so we would get a more accurate model by including time. On the other hand, adding time as a separate component in the state space means that the number of states is increased by a factor of T, where T is the total number of time units that we deem relevant. (For example, if we decide that all that is relevant is the hour of the day, then there would be 24 time units; if we want to model the behavior at the level of minutes, then we would have 1440 time units.) It seems to us that, in practice, the overhead of including time is not worth the ensuing computational complexity, and we are better off having a number of separate MDPs, one for each expected pattern, and solving each of them separately. (Thus, for example, we could have a "mid-day" MDP, a middle-of-the-night MDP, and "morning rush hour" MDP, and so on.)
- Should we include information about neighboring cells in the state? The number of calls in neighboring cells (together with the mobility model) in fact determines the handoff probability (the distribution that describes the probability that a call will be handed off in the next t time units). The handoff probability is in fact necessary for even defining the MDP, since it affects the transition probability. On the other hand, keeping track of this information can significantly increase the size of the state space. Fortunately, in many cases of interest, it is possible to estimate the handoff probability (and thus the transition probability), although subtleties arise that seem to have been ignored in other papers on the subject. We return to this issue in Section IV.
- Should we keep track of how long each call has been in the system? There are (at least) two reasons why such information may be necessary. For one, it may be necessary in order to estimate how much longer the call will remain in the system. (For example, if the length of a call is characterized by a heavy-tailed distribution, then the probability that a call terminates depends on how long it has lasted.) For another, if the utility of a call is a nonlinear function of the length of the call, it will be necessary to include this information in the state in order to correctly calculate the utilities. In our simulations, neither of these reasons apply. Our assumptions guarantee that the probability of a call termination is independent of how long it has been in progress (i.e. the Markovian property) and our utility does not depend in a nonlinear way on the length of the call.
- Do we have to include in the state information about the most recent call (what QoS class it is in, whether it is a new call or a handoff, etc.)? We have decided to include this information; the alternative would be to have a much more complicated choice of actions. We return to this issue in Section III-B.

Given these considerations, we take the state space S to consist of vectors of length $K + 1$. (Recall that K is the number of QoS classes.) We set $s = (x_1, \ldots, x_K, \theta) \in S$, where x_i is the number of ongoing calls in QoS class i, for $i = 1, \ldots, K$, and θ is a description of the *call event* that happens at the current time unit if there is one, and

n (for no event), otherwise. There are three types of call events: a new-call arrival of QoS class i, a handoff-call arrival of QoS class i, and call departure of QoS class i. We describe these events by pairs of the form (r, i), (h, i), and (d, i), respectively. We assume that at most one call event happens at each time. Thus, there are at most $3K+1$ possible values of this last component. (The (d, i) event cannot occur if there are no calls of QoS class i currently in the cell.) The state space is constrained by the total number of channels in the system. We assume that each call in QoS class i requires b_i channels and that there are N channels altogether. Note that "channels" are not necessarily physical channels; they can be logical channels as well (for instance, the number of concurrent users in CDMA). Thus, $S = \{s = (x_1, \ldots, x_K, \theta) : x_i \geq 0,\ \sum_{i=1}^{K} b_i x_i \leq N\}$.

B. The Action Space and Transition Function

Given our representation of states, we need only two possible actions: *accept* and *reject*. These actions have the obvious effect if there is a call arrival at the current time unit, and no effect at all if there is a call departure or no call event at the current time unit. This is captured by the transition function. Note that the effect of an action on the first k components of a state is completely deterministic; the only uncertainty is what the next call event will be. For example, if $K = 2$, then in state $(2, 3, (r, 2))$, *accept* results in a state of the form $(2, 4, \theta)$ while reject results in a state of the form $(2, 3, \theta)$. Similarly, in a state of the form $(2, 3, (d, 2))$, both accept and reject result in a state of the form $(2, 2, \theta)$. The relative probability of each of these outcomes depends on our assumptions about dwell times, mobility (how often calls leave one cell for another), and the probability of call arrival. It may also depend on the policy itself, since the policy affects the number of calls in each cell, which in turn may affect the handoff probability. For example, if the policy rejects all calls, then the handoff probability is guaranteed to be 0. We return to this issue in Sections IV and V.

We can represent the transition function in terms of two matrices, one for the action *accept* and one for *reject*. The (s, t) entry of the *accept* matrix describes the probability of going from s to t if the *accept* action is performed, and similarly for *reject*. Since there are only $3K + 1$ entries in each row that have positive probability, the matrix is relatively sparse. This may make it possible to speed up some of the computations involved.

As we said earlier, we could use a state representation that did not include the last component. We would then need actions of the form $(a_1, \ldots, a_{2K})$, where each a_i is either *accept* or *reject*. For $i = 1, \ldots, K$, the a_i component tells us what to do if the next call event is an arrival of a new call of QoS class i; for $i = K+j$, $j = 1, \ldots, K$, the a_i component tells us what to do if the next call event is a handoff of QoS class j. Of course, if the next call event is a departure, then we again take the obvious transition. In [26], it was suggested that this representation would lead to computational efficiencies. However, since we save a factor of only $3K + 1$ in the size of the state space while increasing the number of possible actions by a factor of 2^K, and since the standard algorithm runs in time $|S|^2|A|$, it seems unlikely that this approach would indeed be more computationally efficient.

C. The Reward Function

The reward function makes use of the utilities of the QoS classes. However, these utilities do not completely determine the reward. For example, if the utility of QoS class i is u_i, do we obtain the utility (i.e., reward) only once (say when the call is connected)? Do we obtain it for each unit of time that the call is connected? The first possibility corresponds to a charge of a flat rate per call (with perhaps some penalties for dropping a call or blocking it); the second corresponds to a charge per call that is a linear function of its duration. Clearly other schemes are possible as well. Our approach can easily accommodate both flat-rate pricing and linear pricing in a natural way. We represent the reward R by a matrix (R_{ij}), $i = 1, \ldots, K$, $j = 0, 1, 2$. Roughly speaking R_{i0} is the reward for accepting a call of QoS class i, if we are thinking of flat-rate pricing, and the reward for carrying a call of QoS class i for a unit of time, if we are thinking of linear pricing. R_{i1} is the penalty for blocking a call of QoS class i and R_{i2} is the penalty for dropping a call of QoS class i.

With the reward matrix in hand, we can now describe the reward function $R(s, a)$ in a straightforward way for both flat-rate pricing and linear pricing. (We do not consider other reward policies here.) For flat-rate pricing, we have:

- $R((\vec{x}, \theta), \textit{accept}) = R_{i0}$ if $\theta = (r, i)$
- $R((\vec{x}, \theta), \textit{accept}) = 0$ if θ is (d, i), (h, i), or n
- $R((\vec{x}, \theta), \textit{reject}) = R_{i1}$ if θ is (r, i)
- $R((\vec{x}, \theta), \textit{reject}) = R_{i2}$ if θ is (h, i)
- $R((\vec{x}, \theta), \textit{reject}) = 0$ if θ is (d, i) or n.

For linear pricing, we have:

- $R((\vec{x}, \theta), \textit{accept}) = R_{i0} + \sum_{j=1}^{K} x_j R_{j0}$ if θ is (r, i) or (h, i)
- $R((\vec{x}, \theta), \textit{accept}) = \sum_{j=1}^{K} x_j R_{j0}$ if θ is (d, i) or n
- $R((\vec{x}, \theta), \textit{reject}) = R_{i1} + \sum_{j=1}^{K} x_j R_{j0}$ if θ is (r, i)
- $R((\vec{x}, \theta), \textit{reject}) = R_{i2} + \sum_{j=1}^{K} x_j R_{j0}$ if θ is (h, i)
- $R((\vec{x}, \theta), \textit{reject}) = -R_{i0} + \sum_{j=1}^{K} x_j R_{j0}$ if θ is (d, i)
- $R((\vec{x}, \theta), \textit{reject}) = \sum_{j=1}^{K} x_j R_{j0}$ if θ is n.

IV. Computing the Optimal Policy

Our goal is to find the policy (mapping from states to actions) that maximizes the average sum of rewards. The optimal policy can be obtained using dynamic programming, by a modification of standard techniques like value iteration or policy iteration (see [24] for more details). To explain why we need to modify the standard techniques, we first briefly review the value iteration algorithm (although the points we make apply equally well to policy iteration, the other standard approach, and all their variants). The value iteration approach is based on the following observation. Let p^a_{xy} be the probability of making the transition from state x to state y if action a is taken (this probability is defined by the transition function). Suppose π is some policy for the MDP. Let $V_\pi(x)$ be the value of state $x \in S$ for policy π, that is, the expected reward if we use policy π starting in state x. The idea behind value iteration is that we can compute the optimal policy π^* and the optimal value function V^* by successive approximations. We start with an arbitrary value function V_0. Let π_0 be the optimal choice of action with respect to V_0; that is

$$\pi_0(x) = \mathrm{argmax}_{a \in A} \left(R(x,a) + \sum_{y \in S} P^a_{xy} V_0(y) \right), \quad (1)$$

where P^a_{xy} is the probability $T(x,a)(y)$ of making the transition from x to y if action a is chosen. Suppose we have defined $\pi_0, \ldots, \pi_n$ and $V_0, \ldots, V_n$. We then define V_{n+1} and π_{n+1} as follows:

$$V_{n+1}(x) = \max_{a \in A} \left(R(x,a) + \sum_{y \in S} P^a_{xy} V_n(y) \right); \quad (2)$$

$$\pi_{n+1}(x) = \mathrm{argmax}_{a \in A} \left(R(x,a) + \sum_{y \in S} P^a_{xy} V_{n+1}(y) \right). \quad (3)$$

It is a standard result that π_{n+1} converges to an optimal policy π^* and V_n converges to the value V^* of π^*. In practice, we choose some ϵ and stop the computation when $|V_n(x) - V_{n+1}(x)| < \epsilon$ for all states $x \in S$; π_n is then an acceptable approximation to π^*.

We want to apply value iteration to computing optimal admission decisions. But, as we hinted above, there may be a problem even defining the MDP. Admission decisions in cell c must take into account how many calls will be handed off from neighboring cells. If many more handoff calls are likely to arrive, the admission decision must make new call admission decisions more conservatively. How many handoff calls will arrive at a cell c depends on the number of calls in c's neighbors and on the mobility pattern. However, if c's state does not include the number of calls in neighboring cells (as is the case in our model), we must find some way of estimating the number of calls at c's neighbors in order to estimate the handoff probability (which in turn affects the transition probability in the MDP). There are two (conflicting) intuitions regarding this estimate. The first is that the number of calls in the current cell (which is part of the cell's state) is a good estimate of the number of calls at its neighbors. Note that this says that the number of calls in a cell and its neighbors is strongly correlated. The second intuition suggests that, given a policy, a good estimate of the number of calls in each QoS class at a neighboring cell is just the expected number of calls in each QoS class in a state over time as the policy is run. This intuition suggests that, given a policy, the number of calls at c's neighbors is uncorrelated with the number of calls at c.

If the first intuition is correct (as we expect it will be in some cases), then it is relatively straightforward to determine the probability that a call will be handed off to a cell c in state s (given the mobility model) and thus to define the transition probabilities for the MDP. However, if the second intuition is correct (as experimental evidence shows that it is under the assumptions of our simulation) then the handoff probability (and hence the transition probability) depends on the policy. Thus, it seems we cannot even define the MDP, let alone use value iteration to compute the optimal policy! (We remark that this problem seems not to have been noticed in other papers that use MDPs in this context [25], [28], [33], which simply assume that the handoff probability is fixed, independent of the policy, and is given as part of the model.)

Fortunately, even if the second intuition is more appropriate, it is still often possible to find the optimal policy by a relatively simple modification of value iteration. We start by guessing a vector $\vec{c}_0$ that describes the expected number of calls in each QoS class. Under the second intuition, provided the guess is correct, it (along with the known call arrival probability and mobility model) is enough to determine the handoff probability and thus the transition probabilities, and hence define an MDP. We then use standard value iteration to obtain the optimal policy π^*_0 for the resulting MDP. Under minimal assumptions (namely, that for any two states s and s', the probability of reaching s' starting in s is positive when using policy π^*_0, which is certainly the case for this problem), the policy π^*_0 determines a *stationary probability distribution* P_0 over states [24]. The probability of a state s according to P_0 can be viewed as the probability of finding a cell in state s if we sample at random. Let $\vec{c}_1$ be the expected number of calls of each QoS class according to P_0. We then use $\vec{c}_1$ to calculate the handoff probability, and thus determine the transition probabilities for a new MDP. We can then calculate the optimal policy π_1^* for this MDP. We then iterate this procedure. In general, it seems that this approach should converge under some reasonable assumptions, but we have not yet proved this analytically. It does converge for our simulation. However, even if it converges, there is no guarantee that it will converge to an optimal MDP. All that we can guarantee in general is that it converges to a local optimum. Methods such as simulated annealing [14], genetic algorithms [7], or Tabu search [6] should be useful for finding a global optimum.

V. Experimental Results

We have compared our MDP approach to one well-known heuristic in the literature: the NAG policy [4], [21]. We have chosen NAG because it is reported in [4] to be one of the best admission-control policies in terms of balancing utilization efficiency and connection-level QoS.

A. Experimental assumptions

In performing our experiments, we made a number of assumptions. To make the simulations easier to run, we assumed that there are only two QoS classes, one for data and one for video. We used the mobility model proposed in [11], extended to account for multiple QoS classes and the finite capacity of the cell BS. We also made the following traffic and mobility assumptions:

- The region is covered by 19 hexagonal cells, each cell with a radius $R = 1Km$. In order to simulate a large area, when a call is handed off beyond the region boundary, it is handed back to another cell on the boundary (with cells having fewer neighbors having a proportionately higher chance of being chosen). This adjusts for the fact that, otherwise, the number of handoffs received on average at the boundary cells will be smaller than at the interior cells.
- Mobiles can travel in any direction with equal probability and with a constant speed SP.
- Each mobile goes through changes of direction at "arbitrary" places and times.
- Connection requests are generated according to a Poisson process with rate λ (connections/second/cell) in each cell.
- Each connection can be a data call with probability R_{dv} or a video call with probability $1 - R_{dv}$.
- The call-holding time distribution is the same for both video and data, and is exponentially distributed with mean 120 seconds ($\frac{1}{\mu} = 120$ seconds).
- The dwell time is exponentially distributed with mean $\frac{1}{\rho\mu}$ seconds, where $\rho = \frac{(3+2\sqrt{3})SP}{9\mu R}$.
- A video call occupies 4 BU, where a BU is defined to be the bandwidth to carry out one voice call.
- Each cell has a fixed capacity of 100 BU.

Identical assumptions have been made in many similar studies [16], [15], [17], [18], [21], [27], [28], [31], [32], [33] so, for the sake of comparison, we make them here as well. While these assumptions seem quite reasonable in many contexts, we are well aware that they may be quite inappropriate in others. For example, if we are considering traffic along a highway, assuming that the mobile's movement is random is clearly incorrect, although it may be more reasonable if we are considering calls in midtown Manhattan. The assumption that call-holding times are exponentially distributed becomes less appropriate as wireless traffic starts to approximate Internet traffic; studies of the Internet suggest that heavy-tailed distributions are more appropriate [22], [23], [30]. Indeed, recent studies of telephone traffic suggest that even for telephone traffic, heavy-tailed distributions may be appropriate [5], although work by Guerin [11] supports the use of the exponential distribution. In any case, we stress that we are making these assumptions only for our case study. Nothing in our general framework depends on them.

Under the assumptions made here, the modeling approach we used in Section III is appropriate. Our traffic and mobility model do not depend on time. In addition, our assumptions about Poisson arrival process and exponential call holding time guarantee the *Markov* property, so we do not need to put time into the state space. We consider flat rate and linear rate pricing; therefore, we do not keep track of how long each call has been in the system. We assume that traffic patterns are homogeneous. The state of the neighboring cell affects the current cell only through the arrival rate of handoff calls, which depends on the number of calls in the cell. Since experimental evidence shows that, under our assumptions, the number of calls at a neighbor is the expected number of calls in a state over time as the policy is run (i.e., the second intuition holds), we can compute the handoff probability and the optimal policy by using the modified value iteration approach discussed in Section IV.

In applying this approach, it is necessary to calculate the transition probabilities for the MDP given an assumption $\vec{c}$ about the expected number of calls of each class. As we observed earlier, all that is needed is to compute the relative probability of the possible types of next call event; the detailed calculations can be found in Appendix A. We now consider the issue of convergence. Under the assumptions used in our simulation, the optimal policy is a threshold [20]. That is, for each type of call-arrival event, there is a threshold such that, if there are fewer calls currently in the system than the threshold, the call is admitted; otherwise it is rejected. If there is only one QoS class, this modification of value iteration converges and computes the optimal policy. The reason for this convergence result is as follows: It is clear that the higher the handoff probability the lower the acceptance threshold (that is, the lower the threshold t such that calls are rejected if there are more than t calls already in state t), and vice versa. We can find the optimal policy by doing a binary search; that is, start with an assumption c_0 about the expected number of calls; compute the optimal policy π_0^* according to c_0, and compute the expected number c_0' of calls corresponding to π_0^*. By the argument above, the expected number of calls under the true optimal policy is between c_0 and c_0'. Let $c_1 = (c_0 + c_0')/2$. We compute the optimal policy with respect to c_1, then iterate. Although this argument does not work in higher dimensions, nevertheless, our simulations do always converge.

B. The NAG Policy

The design goal of NAG is to keep the handoff-dropping probability P_{hd} below a certain threshold α, thereby maintaining connection-level QoS. A new call is admitted at cell c if admitting it does not raise the handoff-dropping probability of existing calls in the system above α and, if admitted, the new call's handoff-dropping probability is also below α. The problem is to decide the effect on P_{hd} of accepting the new call. NAG does this by estimating the state of cell c and its neighboring cells T_{est} units of time after the new-call arrival time, for some appropriately chosen T_{est}. The choice of T_{est} is based on the current state of each cell and the aggregate history of handoffs observed in each cell. As observed in [4], the performance of NAG depends critically on the choice of the interval T_{est}. It is assumed that T_{est} is small enough that the probability that a call experiences a handoff more than once during T_{est} time units is negligible. We experimented with different values of T_{est} and chose it to be 5 seconds, since this choice seemed to give the best results for NAG in our experiments.

C. Numerical Results

There are two main issues we want to study in our simulation. First, we want to understand the behavior of the optimal policy, that is, what it depends on and what it does not depend on. Thus, we study the effect of user utility, the pricing scheme, and traffic and mobility patterns on the optimal policy. Second, we want to compare the performance improvements of the optimal policy over NAG under various conditions. The comparison will yield insight into the behavior of NAG and provide guidelines for the design of more efficient heuristic admission-control policies.

For these studies, we set the reward matrix R_{ij} (see Section III) as follows. We took R_{i0} to be proportional to the bandwidth requirement of each QoS class; we set R_{i1} to be negative and its absolute value to be 10% of R_{i0}. We varied the ratio r_{db_i} (R_{i2} divided by R_{i1}) to study its effects on the optimal policy. We considered two values for R_{dv}: 1 (which means that all calls are data calls) and 0.5 (which means that half the calls are data and half are video). We also considered two settings for the average speed of the mobile: 100 Km/hr and 50Km/hr. We refer to the former setting as the high user mobility case and the latter as the lower user mobility case.

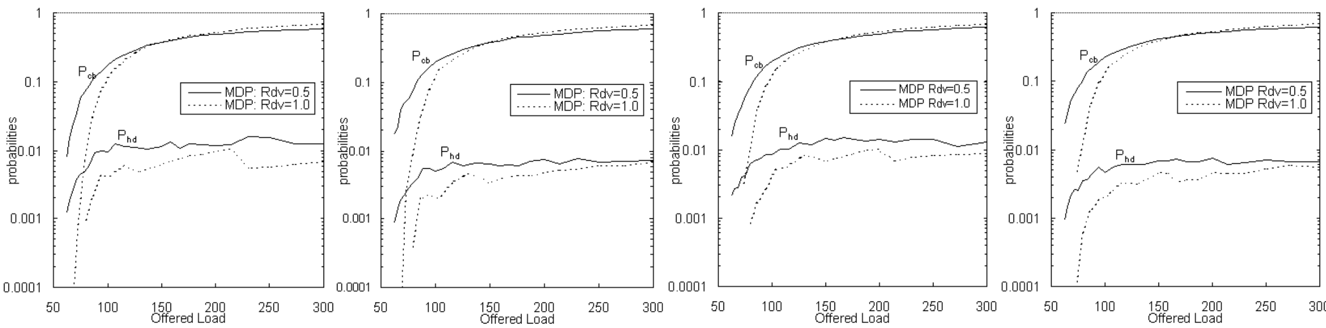
C.1 The Characteristics of the Optimal Admission Policy

First, we want to study the effect of user utility on the optimal admission policy. To do so, we vary r_{db_i} (the ratio between call dropping and call blocking penalty) for the two QoS classes. In Figure 1, we see that for the same offered load, the call-dropping probability in the case of $r_{db_1} = r_{db_2} = 80$ is smaller than the call-dropping probability in the case of $r_{db_1} = r_{db_2} = 40$ case. This is what we would expect, since the optimal admission-control policy takes the application utilities into account. The greater the relative penalty for dropping a call, the lower the probability that it will be dropped. Figure 1 also shows that the optimal admission policy does not depend strongly on the traffic pattern parameter R_{dv} or on the mobility parameter SP.

Our simulations do show that the pricing scheme can have a significant impact on the optimal control policy. If $r_{db_1} = r_{db_2} = 80$, the call dropping probability for the flat pricing scheme is around 1% under high load (see Figure 1(b)), but for the linear pricing scheme, the call dropping probability is around 10% under the same high load (see Figure 2) . This result is also very intuitive. Since the reward ($R_{i0} \times 1/\mu$) for accepting a call in the linear pricing scheme is much higher than the reward (R_{i0}) for accepting a call in the flat pricing scheme, the linear pricing scheme tends to accept more calls than the flat pricing scheme.

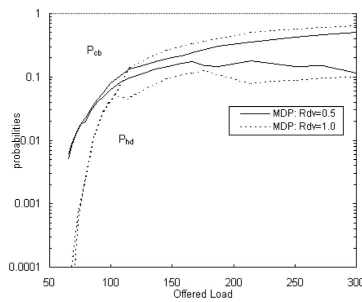
C.2 The Expected Utility of Different Admission Control Policies

The goal of the network service provider is to maximize its revenue while providing satisfactory service to the users over time. So the network performance should be measured in terms of the average system revenue over time. Since the system's utility reflects the utility of users (to some extent), the average system revenue over time also reflects how good is the service that the system provides to the users over time. In our simulation, we compare the expected utilities of NAG and the optimal control policy



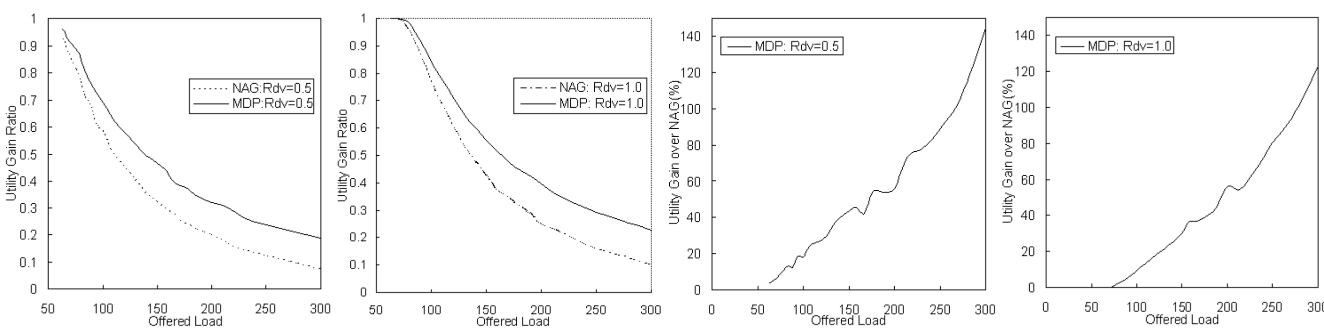
(a)SP=50: $r_{db_1} = r_{db_2} = 40$ (b)SP=50: $r_{db_1} = r_{db_2} = 80$ (c)SP=100: $r_{db_1} = r_{db_2} = 40$ (d)SP=100: $r_{db_1} = r_{db_2} = 80$

Fig. 1. P_{cb} and P_{hd} vs. offered load: optimal admission-control policy (flat pricing scheme).



$r_{db_1} = r_{db_2} = 80$

Fig. 2. P_{cb} and P_{hd} vs. offered load: optimal admission-control policy (linear pricing scheme).



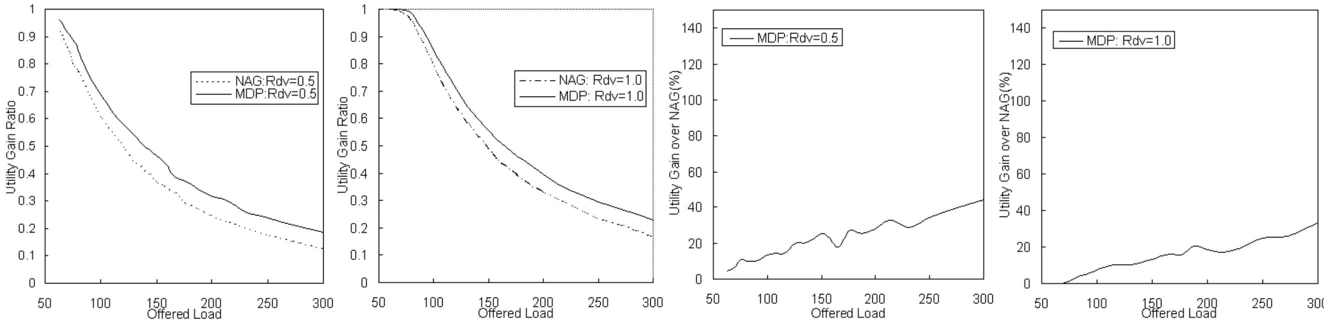
(a) $R_{dv} = 0.5$ (b) $R_{dv} = 1.0$ (c) $R_{dv} = 0.5$ (d) $R_{dv} = 1.0$

Fig. 3. $r_{db_1} = r_{db_2} = 80$, NAG's design goal is $P_{hd} = 4\%$. (a),(b): Expected utility ratio of MDP and NAG; (c),(d): MDP's utility gain over NAG.

under different offered loads. To capture the performance degradation of the network as the load increases, the expected utility is normalized with respect to a network with infinite capacity. In this section, we present the results for the flat pricing scheme and lower mobility case. Similar results hold for the linear pricing scheme and for high mobility; we omit details here.

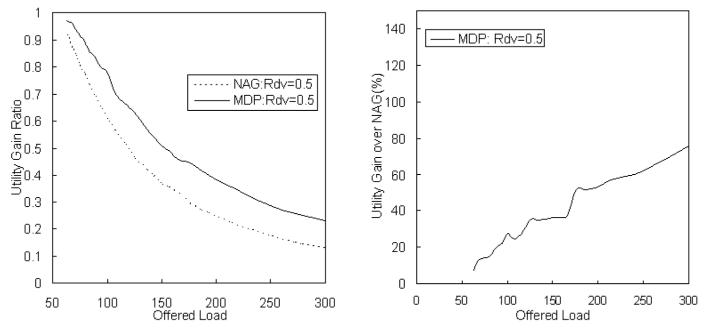
Note that our optimal admission-control policy takes the user's utility into account automatically. However, NAG does not. The call-dropping probability is selected by NAG in an *ad hoc* way. With the right choice of parameters, the performances gap between our optimal admission-control policy and NAG decreases. On the other hand, with the wrong choice, the opposite is true.

As our experimental results show, there is no one setting of the call-dropping probability that gives good results over the whole range of interest. For example, Figure 3 describes the results when NAG picks $P_{hd} = 4\%$ as the design goal. The optimal admission-control policy achieves a performance improvement of approximately 18%, 55% and 144% over NAG at offered loads of 100, 200, and 300, respectively, in the $R_{dv} = 0.5$ case (see Figure 3(c)). (At load 300, the improvement of 144% corresponds to a factor of more than 3 in the call-dropping probability



(a) $R_{dv} = 0.5$ (b) $R_{dv} = 1.0$ (c) $R_{dv} = 0.5$ (d) $R_{dv} = 1.0$

Fig. 4. $r_{db_1} = r_{db_2} = 80$, NAG's design goal is $P_{hd} = 1\%$. (a),(b): Expected utility ratio of MDP and NAG; (c),(d): MDP's utility gain over NAG.



(a) $R_{dv} = 0.5$ (b) $R_{dv} = 0.5$

Fig. 5. $r_{db_1} = 2$, $r_{db_2} = 80$, NAG's design goal is $P_{hd} = 1\%$. (a): Expected utility ratio of MDP and NAG; (b): MDP's utility gain over NAG.

while maintaining almost the same call blocking probability.) With $P_{hd} = 1\%$ as NAG's design goal, as shown in Figure 4, the optimal admission-control policy achieves a performance improvement of approximately 14%, 21%, and 40% over NAG at offered loads of 100, 200, and 300, respectively, in the $R_{dv} = 0.5$ case. The improvement in the normalized expected utility depends in part on the type of calls that must be dealt with. For example, the improvements in terms of normalized expected utility over NAG at offered load 100, 200, and 300 is 11%, 44%, and 122%, respectively, in the case of $R_{dv} = 1.0$ when $P_{hd} = 4\%$ is the design goal (see Figure 3(d)). This is because the optimal policy is biased towards data calls, since data calls can still be accepted in states that video call cannot be accepted due to insufficient bandwidth.

The differences between NAG and our optimal MDP policy become even more dramatic if we change r_{db_1} from 80 to 2 (making the penalty for dropping a data call much closer to that of blocking a call; as we observed in the introduction, this might make sense for FTP transfers). In this case, the improvement of the MDP approach over the NAG approach increases from 40% to 70% at offered load 300, with $P_{hd} = 1\%$ as NAG's design goal (see Figure 5)
.

Of course, once we add utilities into the picture, it would not be difficult to modify NAG so that it attempted at all times to adjust P_{hd} to obtain optimal performance. But by doing this, NAG would essentially be approximating our MDP approach. In fact, as our state space grows, we will need to find useful approximation to the MDP approach, since it becomes computationally too expensive. NAG may be one such approach, but there may be better ones as well. The MDP approach at least gives us a framework by which to judge this issue.

VI. Discussion and Related Work

There has been a great deal of related work on admission control, both in the context of wireless and wired networks. Optimal call-admission policies in cellular networks which carry only voice calls were studied in [25], [27]. Near-optimal admission-control policies [31], [32] and heuristic approaches [3], [21] have been studied. Bandwidth adaptation using layered encoding has been studied recently in [15], [16], [17]. None of these papers tried to provide a general decision-theoretic framework for the call-admission problem. Rather, they typically focused on the standard scenario (Poisson arrival times; exponentially distributed call holding and dwell times) and tried to solve the problem in that setting. While we believe that our main contribution is the general framework, not its application to the standard scenario, as we saw, our approach does dominate a well-known heuristic approach in this setting.

Previous work has either had the goal of keeping the call-dropping probability below a given threshold [3], [21] or used a weighted factor between dropping and blocking as the design goal [25], [27], [31], [32]. The problems with choosing an appropriate handoff-dropping probability threshold have already been discussed in detail. The second approach can be viewed as a special case of ours. In [25], [27], [31], [32] the weighting factor is determined by the relative utilities of call blocking and call dropping. Thus, these approaches are essentially a special case of ours, using flat-rate pricing, with only one QoS class. They assume that the call handoff rates are fixed, and do not deal with the fact that they are influenced by the policy. In addition, these papers do not address subtleties such as the effect of the policy chosen on the handoff rates.

There is another large body of literature on admission control in wired networks. The measurement-based admission-control (MBAC) approach [9], [10], [12], [29]. [8] is perhaps most relevant to our work. For example, in [8], Renegotiated Constant Bit Rate (RCBR) Service is proposed for variable bit rate video. The main idea to increase the throughput through statistical multiplexing while providing delay and loss guarantees through renegotiation. While the basic idea of renegotiation may prove applicable in the wireless setting, we cannot immediately use this approach due to significant differences between the wired and wireless settings. In particular, since a wireless channel can be used exclusively by only one flow at a time, we cannot multiplex many flows on a wireless channel as easily as we can multiplex flows on bandwidth in the wired case. (For example, if both flows transmit at the same time, the data will be garbled in the wireless case.) Moreover, in MBAC, even if renegotiation fails, the flow keeps its original bandwidth. But for the wireless counterpart, if a handoff fails, the flow loses its original bandwidth.

Our general approach leads to some interesting research issues. All our algorithms are polynomial in the size of the state space. In our model, we used a number of techniques to limit the size of the state space. However, in realistic cases, we can expect to have much larger state spaces. Although the computation of the optimal solution is done off-line, and the admission decisions are made simply through table lookup, there is a limit to the number of states we can feasibly handle. We are currently exploring techniques to provide good approximate solutions in the presence of large state spaces. Motivated by the result in [13] that the running time in computing the near-optimal policies in arbitrarily large, unstructured MDPs does not depend on the number of states, we believe that MDP can be quite applicable in practice. We expect that, if the state space is large (due to many QoS classes) the behavior of the system will be qualitatively similar to the case when it is small.

References

[1] T. F. Abdelzaher and K. G. Shin. End-host architecture for qos-adaptive communication. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, pages 121–130, 1998.

[2] E. Amir, S. McCanne, and M. Vetterli. A layered dct coder for internet video. In *Proceedings of the IEEE International Conference on Image Processing*, 1996.

[3] S. Choi and K. G. Shin. Predictive and adaptive bandwidth reservation for hand-offs in qos-sensitive cellular networks. In *Proceedings of the ACM SIGCOMM*, pages 155–166, 1998.

[4] Sunghyun Choi and Kang G. Shin. Comparison of connection admission-control schemes in the presence of hand-offs in cellular networks. In *Proceedings of the 4th ACM International Conference on Mobile Computing and Networking*, 1998.

[5] D. E. Duffy, A. A. McIntosh, M. Rosenstein, and W. Willinger. Statistical analysis of ccsn/ss7 traffic data from working ccs subnetworks. *IEEE Journal on Selected Areas in Communications*, 12(3):544–551, 1994.

[6] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[7] D.A. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, MA, 1989.

[8] M. Grossglauser, S. Keshav, and D. Tse. Rcbr: A simple and efficient service for multiple time-scale traffic. In *Proceedings of ACM SIGCOMM*, 1995.

[9] M. Grossglauser and D. Tse. A framework for robust measurement-based admission control. In *Proceedings of ACM SIGCOMM*, 1997.

[10] M. Grossglauser and D. Tse. A time-scale decomposition approach to measurement-based admission control. In *Proceeding of IEEE INFOCOM*, 1999.

[11] R. Guerin. Channel occupancy time distribution in a cellular radio system. *IEEE Transaction on Vehicular Technology*, 35(3):89–99, 1987.

[12] Sugih Jamin, Peter Danzig, Scott Shenker, and Lixia Zhang. A measurement-based admission control algorithm for integrated services packet networks. *IEEE/ACM Transactions on Networking*, 5(1):56–70, 1997.

[13] M. Kearn, Y. Mansour, and A. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *The International Joint Conferences on Artificial Intelligence*, 1999.

[14] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598)(5):671–680, 1983.

[15] T. Kwon, Y. Choi, C. Bisdikian, and M. Naghshineh. Measurement-based call admission control for adaptive multimedia in wireless/mobile networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 1999.

[16] Taekyoung Kwon, Jihyuk Choi, Yanghee Choi, and Sajal Das. Near optimal bandwidth adaptation algorithm for adaptive multimedia services in wireless/mobile networks. In *IEEE Transaction on Vehicular Technology*, September 1999.

[17] Taekyoung Kwon, Sajal Das, Ilkyu Park, and Yanghee Choi. Bandwidth adaptation algorithms with multi-objectives for adaptive multimedia services in wireless/mobile networks. In *ACM WoWMoM*, August 1999.

[18] S. Lu and V. Bharghavan. Algorithms for resource management in indoor mobile computing environments. In *Proceedings of ACM SIGCOMM*, pages 231–242, 1996.

[19] S. McCanne. *Scalable Compression and Transmission of Internet Multicast Video*. PhD thesis, University of California Berkeley, 1996.

[20] B.L. Miller. A queueing reward system with several customer classes. *Management Science*, 16(3):234–245, 1969.

[21] M. Naghshineh and M. Schwartz. Distributed call admission control in mobile/wireless networks. *IEEE Journal on Selected Areas in Communications*, 14(4):711–717, 1996.

[22] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[23] V. Paxson and S. Floyd. Why we don't know how to simulate the internet. *Proceedings of the 1997 Winter Simulation Conference*, 1997.

[24] M.L. Puterman. *Markov Decision Processes-Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York,NY, 1994.

[25] R. Ramjee, R. Nagarajan, and D. Towsley. On optimal call admission control in cellular networks. *Wireless Networks Journal*, 3(1):29–41, 1997.

[26] K.W. Ross and D.H.K. Tsang. Optimal circuit access policies in an isdn environment: A markov decision approach. *IEEE Transactions on Communications*, 37:934–939, 1989.

[27] M. Saquib and R. Yates. Optimal call admission to a mobile cellular networks. In *Proceedings of IEEE Vehicular Technology Conference*, pages 190–194, 1995.

[28] S. Tekinary and B. Jabbari. A measurement based prioritization scheme for handovers in cellular and microcellular networks. *IEEE Journal on Selected Areas in Communications*, 10(8):1343–1350, 1992.

[29] D. Tse and M. Grossglauser. Measurement-based call admission control: Analysis and simulation. In *Proceedings of IEE Infocom*, 1997.

[30] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5:71–86, 1997.

[31] A. Yener and C. Rose. Near-optimal call admission policies for cellular networks using genetic algorithms. *Proceeding of Wireless*, 1994.

[32] A. Yener and C. Rose. Genetic algorithms applied to cellular call admission problem: local policy. *IEEE Transaction on Vehicular Technology*, 46(1):72–79, 1997.

[33] C.H. Yoon and K. Un. Performance of personal portable radio telephone systems with and without guard channels. *IEEE Journal on Selected Areas in Communications*, 11(6):911–917, 1993.

APPENDIX

I. DERIVING THE STATE-TRANSITION PROBABILITIES

Given our experimental assumptions, the call holding times are exponentially distributed and the call arrival times are determined by a Poisson distribution. [11] shows experimentally the handoff probability is also exponentially distributed; thus, we make that assumption here, and calculate the parameters experimentally, for each assumption $\vec{c} = (c_1, c_2)$ of the expected number of call of each QoS class.

By taking the minimum of all these distributions, we can compute the PDF describing the time that the next event happens in state $x = (x_1, x_2, \theta_x)$. This PDF is defined as

$$f(t) = \omega e^{-\omega t}, \tag{4}$$

where $\omega = (x_1+x_2)\mu+\lambda_1+\lambda_2+(c_1+c_2)\rho\mu$, $\lambda_1 = R_{dv}\lambda$, $\lambda_2 = (1 - R_{dv})\lambda$;

So the expected time until a new event occurs when in state x is

$$\begin{aligned} \tau(x) &= \int_0^\infty t f(t) dt \\ &= \left[\sum_{i=1}^{2} (x_i\mu_i + \lambda_i + c_i\rho\mu)\right]^{-1} \end{aligned} \tag{5}$$

The probability of a transition from state x to state y given action a is:

$$P_{xy}^a = \begin{cases} \lambda_1\tau(x) & y = (x_1+\sigma_1, x_2+\sigma_2, (r,1)) \\ \lambda_2\tau(x) & y = (x_1+\sigma_1, x_2+\sigma_2, (r,2)) \\ c_1\rho\mu\tau(x) & y = (x_1+\sigma_1, x_2+\sigma_2, (h,1)) \\ c_2\rho\mu\tau(x) & y = (x_1+\sigma_1, x_2+\sigma_2, (h,2)) \\ x_1\mu\tau(x) & y = (x_1+\sigma_1, x_2+\sigma_2, (d,1)) \\ x_2\mu\tau(x) & y = (x_1+\sigma_1, x_2+\sigma_2, (d,2)) \end{cases} \tag{6}$$

where

$$\sigma_i = \begin{cases} 1 & \text{if } a = accept \text{ and } \theta_x = (r,i) \text{ or } (h,i) \\ -1 & \text{if } \theta_x = (d,i) \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$