Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2006, Article ID 15876, Pages 1–11
DOI 10.1155/WCN/2006/15876

Convergence in the Calculation of the Handoff Arrival Rate: A Log-Time Iterative Algorithm

Dilip Sarkar,[1] Theodore Jewell,[2] and S. Ramakrishnan[3]

[1] *Department of Computer Science, University of Miami, Coral Gables, FL 33124, USA*
[2] *The Taft School, Watertown, CT 06795-2100, USA*
[3] *Department of Mathematics, University of Miami, Coral Gables, FL 33124-4250, USA*

Received 23 March 2005; Revised 23 August 2005; Accepted 17 October 2005

Recommended for Publication by Vincent Lau

Modeling to study the performance of wireless networks in recent years has produced sets of nonlinear equations with interrelated parameters. Because these nonlinear equations have no closed-form solution, the numerical values of the parameters are calculated by iterative algorithms. In a Markov chain model of a wireless cellular network, one commonly used expression for calculating the handoff arrival rate can lead to a sequence of oscillating iterative values that fail to converge. We present an algorithm that generates a monotonic sequence, and we prove that the monotonic sequence always converges. Lastly, we give a further algorithm that converges logarithmically, thereby permitting the handoff arrival rate to be calculated very quickly to any desired degree of accuracy.

1. INTRODUCTION

Wireless cellular networks provide service to mobile terminals, which can move from a given cell to any adjacent cell multiple times during the lifetime of a particular call. Therefore, a wireless network must take into account the rate at which ongoing calls arrive from neighboring cells, in addition to the arrival rate of new calls. When a user crosses the boundary from one cell to another, the network must react by handing off the call. However, there must be a channel available in the new cell for that call, or else the handoff fails and the service is abruptly terminated.

One approach for improving the likelihood that a free channel is available when a handoff call arrives is the dedication of a certain number of channels in each cell purely for handoff calls. These dedicated channels are called *guard channels*, and earlier works have focused on the benefit of determining the number of guard channels dynamically.

Models of cellular networks are very important for design as well as operation of the network. During the operation of a network, performance parameters can be estimated empirically by collecting data while the network is in operation. In fact, most of the current networks collect performance data and use it for decision making. However, if a network's performance is outside the desired range, some of the control parameters will need adjustment. The amount of adjustment to be made is determined from a model.

Simulation as well as analytical models are used for designing networks. Simulation models require a long computation time. However, in the absence of analytical models, simulation is the only available tool. Also, simulation models are necessary for the final evaluation of networks designed using approximate analytical models. For instance, even though call holding times and cell dwell times do not follow exponential distributions (see [1, 2]), analytical models assume that they are exponentially distributed. Therefore, a cellular network designed using such a model can be fine-tuned using simulation.

On the other hand, analytical models are computationally efficient. One can estimate performance parameters very quickly. For instance, the *(fuzzy associative memory)* FAM-based call admission controller, reported in [3, 4], used a simulation model for development of the FAM. It took about two months of simulation time on a Pentium IV PC to develop the FAM. However, the FAMs for the call admission controllers reported in [5, 6] were developed using the algorithm presented in this paper requiring about a day on the same Pentium IV PC.

Since the late eighties, the modeling of wireless cellular networks for analysis of their performance has produced sets

of nonlinear equations with interrelated parameters. These nonlinear equations have no closed-form solution, so the numerical values of the parameters are calculated by iterative algorithms. When these iterations fail to converge, however, the precise values of the parameters cannot be determined.

The foregoing applies to wireless cellular networks, for which many studies have used Markov chains as models [7–10]. Some of these models treat all calls identically, while others create a priority status for handoff calls. With respect to the prioritization of handoff calls, there are two basic approaches: (a) the early reservation of channels and (b) the use of guard channels that are dedicated exclusively to handoff calls [8–11].

The number of guard channels can be established in advance (statically) or as an ongoing process (dynamically) (see [12, Chapter 2]). In the former case, bandwidth may be underutilized or handoff call failure rate may be too high. In the latter case, there must be a continual computation of the optimal number of guard channels [7, 11]. This in turn creates a need for the computation of the handoff arrival rate. Note that although *current* handoff call arrival rate can be estimated from some "time averaging" of recent handoff call arrival records, the handoff call arrival rate that would result from the change of the number of guard channels must be determined from simulation *or* analytical model. Since simulations require a long time, analytical models are more desirable.

The absence of a closed-form expression for the handoff arrival rate requires an alternate method, which commonly involves the use of iterative algorithms [7]. One standard formula for the calculation of the handoff arrival rate generates a sequence of approximations that may oscillate around the actual value. When this sequence converges, a result is obtained within any desired degree of accuracy. Convergence is not guaranteed, however, and in that instance the sequence develops a bifurcation and oscillates repeatedly between two values above and below the actual handoff rate value.

In [13], fixed-point iteration for calculating the handoff arrival rate is proposed and used to overcome numerical overflow problems when a cell has a large number of channels. The paper also presents an algorithm for computing the optimal number of guard channels, but the optimization algorithm uses the proposed fixed-point iterative algorithm. The authors of that paper indicate that a proof of convergence of their algorithm is an open problem (last sentence of [13, Section VI]). The iterative algorithm in [7] attempts to avoid any potential nonconvergence by partitioning and bounding the solution interval. However, the process is rather slow—linear with the inverse of the desired accuracy.

In this paper, we present a novel iterative algorithm that always converges and which is logarithmic in nature (thereby assuring a relatively fast convergence). We also present proof of convergence of the algorithm. One can find further details of the work reported here in [14].

1.1. Definitions and notation

It is assumed that each cell in a network has a fixed number of channels, and at any given moment somewhere between none and all of them will be in use. Moreover, the cells are assumed to be identical, that is, the system is homogeneous. Calls arriving into a cell can be from one of two sources: (a) a call that was previously accepted by the network and that is now being handed off from an adjacent cell (a *handoff*) and (b) a brand-new call that has just been received by the cell (a *new call*). Two time frames are relevant. The average length of time that a given call remains active from inception to uninterrupted completion is referred to as the *holding time*, whereas the average amount of time that a call remains in any given cell before departing is the *dwell time*.[1]

Calls depart from a cell for one of two reasons: (a) the mobile terminal moves to an adjacent cell or (b) the customer completes the call and terminates the connection. These departures are distinct from calls that never enter the cell (although there is an attempt to enter). For a new call, if there is no available channel, then the call is simply not accepted. For a handoff, if similarly there is not an available channel, then the handoff fails and the existing call is forced to terminate.

1.2. Organization of the remaining sections

In Section 2, we first give a set of nonlinear equations for the parameters derived from a Markov model of a wireless cellular network. We then present one commonly used expression for iterative calculation of the handoff arrival rate and include an algorithm. Next, we use a straightforward example that shows that the iterations converge with one set of values, but fail to converge when a very slight change is made to one of the parameters. We finish the section by explaining the source of the oscillating nonconvergence and by proposing to use an alternative expression and an accompanying novel algorithm for calculating the handoff arrival rate that always converges. In Section 3, we give a rigorous proof that our novel approach always converges, both for the nonpriority case (no guard channels) and the priority case (a network with guard channels). In Section 4, we take the earlier results and give an algorithm that not only converges, but does so logarithmically. Section 6 contains our concluding remarks.

2. MARKOV MODEL AND CALCULATION OF HANDOFF ARRIVAL RATE

We first refine the definitions of two items and then express the steady state probabilities for a homogeneous cellular wireless network with C channels per cell, of which n are nonguard channels (see [8, 9, 13] for the derivation of the following equations). The offered load and the handoff

[1] Models of wireless networks generally treat calls as arriving in the Poisson process and the holding time and dwell time as being exponentially distributed.

load are more precisely

$$\rho = \frac{\lambda_0 + \lambda_h}{\mu + \eta}, \qquad \rho_h = \frac{\lambda_h}{\mu + \eta}. \tag{1}$$

The steady state probabilities where one or more channels are in use can be split into two portions: (a) states in which any arriving call, whether a new call or a handoff one, will be accepted and (b) states in which only handoff calls will be accepted. These are

$$\begin{aligned} P_j &= \frac{\rho^j}{j!} P_0, \quad \text{for } 0 < j \le n, \\ P_j &= \frac{\rho^n \rho_h^{j-n}}{j!} P_0, \quad \text{for } n < j \le C. \end{aligned} \tag{2}$$

The probability for state 0 (the state in which no channels are in use) is a normalization obtained from the fact that $\sum_{j=0}^{C} P_j = 1$,

$$P_0 = \left[\sum_{j=0}^{n} \frac{\rho^j}{j!} + \sum_{j=n+1}^{C} \frac{\rho^n \rho_h^{j-n}}{j!} \right]^{-1}. \tag{3}$$

The blocking probability of a new call (P_b) and the handoff failure probability of an ongoing call (P_{hf}) are given by

$$P_b = \sum_{j=n}^{C} P_j, \qquad P_{hf} = P_C. \tag{4}$$

2.1. Existence of actual value for λ_h

Before giving an expression for the calculation of the handoff arrival rate (λ_h), we note the possible range of values. Clearly the value cannot be negative, so zero is a lower bound. The quantity ηC is an upper bound, since the rate cannot exceed the number of channels C in the cell divided by the average dwell time $1/\eta$. Also, since a finite, irreducible, positive recurrent Markov chain models a cell, it has a unique stationary distribution, and hence P_b and P_{hf} are uniquely determined. Since a standard expression for the handoff arrival rate is

$$\lambda_h = \frac{\eta(1 - P_b)}{\mu + \eta P_{hf}} \lambda_0 \tag{5}$$

(see [8, 9, 13] for the details), for given values of λ_0, μ, η, C, and n, the handoff rate is determined uniquely by (5). We will denote this unique value by λ_h^*.

Iterative algorithms are typically used for the calculation of the handoff arrival rate, where the value from one iteration is then fed into the equation, thereby producing successive values (see (6)). The hope is that these iterations will converge, but some approaches do not always converge.

2.2. Standard approach

The iterative form is

$$\lambda_h(k+1) = \frac{\eta(1 - P_b(k))}{\mu + \eta P_{hf}(k)} \lambda_0, \tag{6}$$

```
      some small value ϵ > 0
λh := new λh := 0
do { the following steps }
      Step 1: λh := new λh
      Step 2: update values for the offered load ρ and handoff
        load ρh per (1)
      Step 3: update the value of P0 per (3)
      Step 4: update the values of state probabilities P1
        through PC per (2)
      Step 5: update the blocking probability Pb and handoff
        failure probability Phf per (4)
      Step 6: compute the new value for λh, that is, new λh,
        per (6)
while (|λh − new λh|/λh > ϵ)
enddowhile
λh := new λh
```

Algorithm 1: Oscillating algorithm.

where $P_b(k)$ and $P_{hf}(k)$ are the values derived from using $\lambda_h(k)$ in the kth iteration.

An algorithm that incorporates this approach is in Algorithm 1.

We will show that this approach works in some situations, but can also lead to oscillations that do not converge. In our examples, assume that there are twenty channels in each cell, of which four are guard channels, and that for any given call the average duration (holding time) is 120 seconds and the average time in any given cell before departing (dwell time) is twelve seconds. Hence, we have the following values:

$$C = 20, \qquad n = 16, \qquad \mu = \frac{1}{120}, \qquad \eta = \frac{1}{12}. \tag{7}$$

2.3. Works sometimes

Without loss of generality, we will choose zero as the initial value for the handoff arrival rate (i.e., $\lambda_h(0) = 0$). If the new call arrival rate (λ_0) is a relatively low figure, such as 0.1, then using (6) will result in $\lambda_h = 0.899\,027\,7$. Figure 1 shows the plot of the sequence of calculated values for the handoff arrival rate beginning with the initial value of $\lambda_h(0) = 0$. The convergence occurs fairly quickly.

2.4. Can oscillate and not converge

On the other hand, increasing the value for λ_0 (the new call arrival rate) very slightly to 0.12 is sufficient to produce oscillations that do not converge. Once again using the initial value of $\lambda_h(0) = 0$, we obtain from (6) the alternating pair of 1.170 851 55 and 0.712 858 as the calculated values for λ_h. Figure 2 illustrates the oscillations.

(1) Why oscillation occurs

Referring to (6), we see that two variables change with each iteration: (a) the blocking probability $P_b(k)$ and (b)

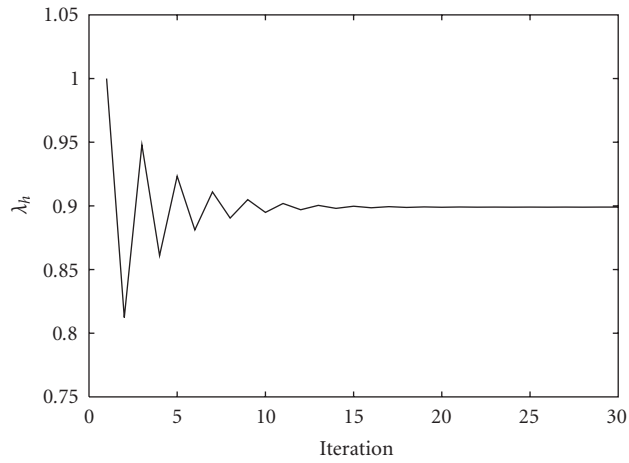

Figure 1: Oscillations that converge.

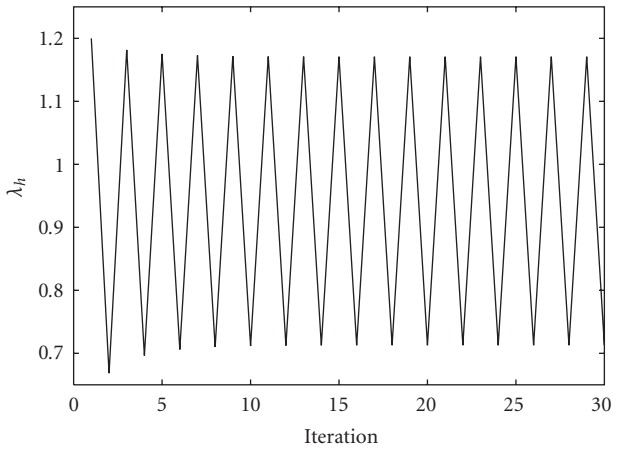

Figure 2: Oscillations that do not converge.

the handoff failure probability $P_{hf}(k)$. The blocking probability is the sum of the steady state probabilities for those states where only handoff calls will be accepted (i.e., states n through C). When $P_b(k)$ is very low, the numerator of (6) becomes larger and results in a higher calculated value for $\lambda_h(k+1)$. The handoff failure probability is the steady state probability for the final state (i.e., state C), and if $P_b(k)$ is low, then so will be $P_{hf}(k)$.

The combination of low calculated values for $P_b(k)$ and $P_{hf}(k)$ produces a higher value for $\lambda_h(k+1)$. When that higher value is then fed into (6), the system's general load ($\rho(k+1)$) and handoff load ($\rho_h(k+1)$) are correspondingly higher. This shifts the weighted average of the state probabilities to the right, with the result that the guard states (states n through C) have higher probabilities. Thus, for this iteration, both $P_b(k+1)$ and $P_{hf}(k+1)$ increase. These increases result in a smaller numerator and larger denominator in (6), thereby producing a smaller calculated value for $\lambda_h(k+2)$ for the next iteration.

This alternation between higher and lower values for the sequence $\lambda_h(k)$ can prevent convergence. The problem occurs when a pair of values produce each other. If we consider (6), and if $x_1 = \lambda_h(k)$ and $x_2 = \lambda_h(k+1)$ represent the values from two successive iterations, the nonconverging oscillation occurs in essence when $f(x_1) = x_2$ and $f(x_2) = x_1$. Figure 3 illustrates this phenomenon.

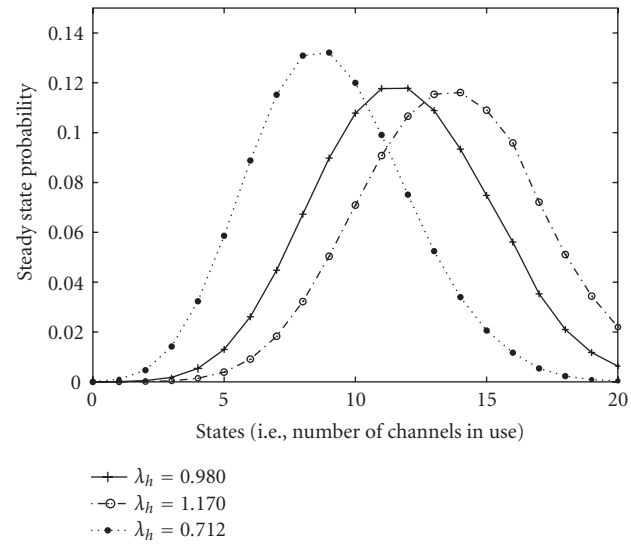

Figure 3: Another view of oscillations that do not converge.

The rightmost plot shows the resulting state probabilities from a value of $\lambda_h(0) = 1.170\,851\,55$. Using these values and values for $P_b(0)$ and $P_{hf}(0)$ in (6) produces a computed value of $\lambda_h(1)$ of 0.712 858. The leftmost plot shows the resulting state probabilities from a value of $\lambda_h(1) = 0.712\,858$. Note that 1.170 851 55 and 0.712 858 are the two nonconverging, alternating values of λ_h illustrated by Figure 2. Consequently, further iterations produce $\lambda_h(2) = \lambda_h(4) = \cdots = \lambda_h(2k) = 1.170\,851\,55$, and $\lambda_h(3) = \lambda_h(5) = \cdots = \lambda_h(2k+1) = 0.712\,858$. Likewise, the computed probability of being in each state alternates from the value in the rightmost plot to the value in the leftmost plot. The third (central) plot shows state probabilities from a value of 0.980 989 06 for λ_h, which is the actual value for λ_h^* (discussed further in the next subsection). *To avoid such a cycle of alternating between two values, what is desired is an iterative algorithm (i) that moves the successive state probabilities monotonically toward their respective steady-state values, and (ii) that moves the successive values of $\lambda_h(k)$ monotonically toward λ_h^*.*

2.5. Avoiding nonconverging oscillations

Rather than using (5) (or its iterative form, (6)) for the calculation of the handoff arrival rate (λ_h), we instead use the basic expression from which (5) is derived (see [8, 13] for details). In general, the value for λ_h is the expected number of channels in use (call it $E(N)$) divided by the average dwell time, that is,

$$\lambda_h = \eta E(N). \tag{8}$$

The value for $E(N)$ is simply the weighted average of the

some small value $\epsilon > 0$
$\lambda_h := \text{new}\,\lambda_h := 0$
do {the following steps}

Step 1: $\lambda_h := \text{new}\,\lambda_h$
Step 2: update values for the offered load ρ and handoff load ρ_h per (1)
Step 3: update the value of P_0 per (3)
Step 4: update the values of state probabilities P_1 through P_C per (2)
Step 5: compute the new value for λ_h, that is, $\text{new}\,\lambda_h$, per (11)

while $(|\lambda_h - \text{new}\,\lambda_h|/\lambda_h > \epsilon)$
enddowhile
$\lambda_h := \text{new}\,\lambda_h$

Algorithm 2: Monotonic algorithm.

number of channels in use:

$$E(N) = \sum_{j=0}^{C} jP_j. \tag{9}$$

Just as there exists a unique steady state value for λ_h given a set of values for the other system components (number of channels, number of guard channels, holding time, dwell time, and new call arrival rate), there is similarly a unique steady state value for $E(N)$.

Combining these ideas, we obtain the following expression for the handoff arrival rate:

$$\lambda_h = \eta \sum_{j=0}^{C} jP_j. \tag{10}$$

The iterative form of the equation is

$$\lambda_h(k+1) = \eta \sum_{j=0}^{C} jP_j(k). \tag{11}$$

The algorithm is identical to the one for the standard approach with the exception that $\text{new}\,\lambda_h$ is calculated using (11) (instead of (6)) and there is now no need to calculate the blocking probability (P_b) or handoff failure probability (P_{hf}).

As with the iterative algorithm using (6), Algorithm 2 is an iterative algorithm where the calculated value for λ_h from one iteration is plugged into the next iteration. In contrast to the use of (6), however, the use of (11) *always converges* and does not experience the oscillations that plagued the first algorithm.

By way of illustration, Figures 4 and 5 show the results from using the set of values for C, n, μ, and η from (7) and the values 0.1 and 0.12, respectively, for λ_0. In both cases we observe convergence, with calculated values of 0.898 920 47[2] and 0.980 989 06 for λ_h.

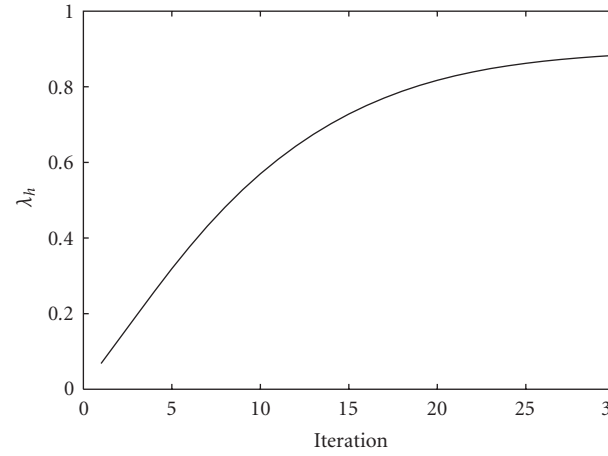

Figure 4: Monotonic: $\lambda_0 = 0.1$.

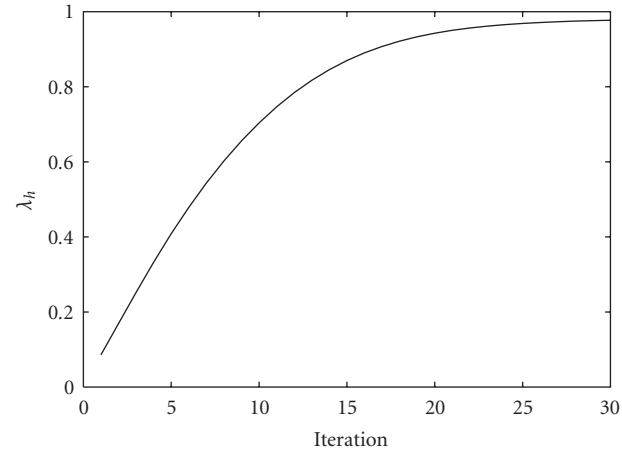

Figure 5: Monotonic: $\lambda_0 = 0.12$.

The reason that the iterative algorithm using (11) always converges is that two things occur simultaneously, and both are monotonic. If we begin with an initial value for $\lambda_h(0)$ that is less than the actual value λ_h^*,

(1) each successive iteration produces values for $E(N)$ and λ_h that are larger than their immediate predecessor values;
(2) no matter how many iterations are done, the calculated values for $E(N)$ and λ_h always remain less than the respective actual values.

If we start with an initial value for $\lambda_h(0)$ that is greater than the actual value, then the reverse holds true (i.e., the

[2] The sharp-eyed reader might detect the slight difference between this value and the one given in Section 2.3. The difference is attributable to the accumulation of round-off errors and does not affect the underlying analysis.

iterations produce successively smaller calculated values for $E(N)$ and λ_h, but always greater than the actual values).

3. CONVERGENCE OF THE MONOTONIC APPROACH

We will denote by $\{P_j(i)\}, 0 \le j \le C$, the steady state probability distribution of the one-dimensional finite, irreducible, positive recurrent Markov chain on $\{0, 1, \ldots, C\}$ determined by the parameters at the ith iteration of the algorithm. Our approach would be to show that these probability vectors satisfy a likelihood ratio ordering, which, as is well known (Lehmann [15, Section 3.3] and Shanthikumar [16]), implies strong stochastic ordering. The special case of this result that we use is stated for ease of reference and completeness.

Lemma 1. *Suppose for nonnegative integers i_1 and i_2,*

$$\frac{P_{j+1}(i_1)}{P_j(i_1)} > \frac{P_{j+1}(i_2)}{P_j(i_2)} \tag{12}$$

for all j, $0 \le j < C$, with the convention that $0/0 = 0$. Then

$$\sum_{j=l}^{C} P_j(i_1) \ge \sum_{j=l}^{C} P_j(i_2) \tag{13}$$

for all l, $0 \le l \le C$, with strict inequality for at least one positive l.

Proof. Let j_0 be the least integer in $\{0, \ldots, C\}$ such that $P_{j_0}(i_1) > P_{j_0}(i_2)$. Such an integer must exist because $\sum_{j=0}^{C} P_j(i_1) = \sum_{j=0}^{C} P_j(i_2) = 1$ and because of the ratio inequality. The remainder of the conditions imply that

$$P_j(i_1) > P_j(i_2) \quad \forall\, j_0 \le j \le C. \tag{14}$$

We cannot have $j_0 = 0$, because $\sum_{j=0}^{C} P_j(i_1) = \sum_{j=0}^{C} P_j(i_2) = 1$. Therefore we have

$$P_j(i_1) \le P_j(i_2) \quad \forall\, 0 \le j < j_0, \tag{15}$$

$$P_j(i_1) > P_j(i_2) \quad \forall\, j_0 \le j \le C. \tag{16}$$

For $l < j_0$, the assertion follows by summing both sides of the inequality (15) over $0 \le j < l$ and subtracting from one. For $l \ge j_0$, the assertion follows by summing both sides of the inequality (16) over $l \le j \le C$. Moreover, we have strict inequality for all $l \ge j_0$. □

Lemma 2. *Suppose for nonnegative integers i_1 and i_2,*

$$\frac{P_{j+1}(i_1)}{P_j(i_1)} > \frac{P_{j+1}(i_2)}{P_j(i_2)} \tag{17}$$

for all j, $0 \le j < C$, with the convention that $0/0 = 0$. Then

$$\sum_{j=0}^{C} jP_j(i_1) > \sum_{j=0}^{C} jP_j(i_2). \tag{18}$$

Proof. By Lemma 1, we have

$$\sum_{j=l}^{C} P_j(i_1) \ge \sum_{j=l}^{C} P_j(i_2) \tag{19}$$

for all l, $1 \le l \le C$, with strict inequality for at least one l. Summing over all l, $1 \le l \le C$, we obtain

$$\sum_{l=1}^{C} \sum_{j=l}^{C} P_j(i_1) > \sum_{l=1}^{C} \sum_{j=l}^{C} P_j(i_2). \tag{20}$$

Interchanging the order of summation yields

$$\sum_{j=1}^{C} \sum_{l=1}^{j} P_j(i_1) > \sum_{j=1}^{C} \sum_{l=1}^{j} P_j(i_2), \tag{21}$$

equivalently,

$$\sum_{j=0}^{C} jP_j(i_1) > \sum_{j=0}^{C} jP_j(i_2). \tag{22}$$

This proves the result. □

Lemma 3 (ratio lemma). *For any nonnegative integers i_1 and i_2,*

$$\rho(i_1) > \rho(i_2) \iff \frac{P_{j+1}(i_1)}{P_j(i_1)} > \frac{P_{j+1}(i_2)}{P_j(i_2)}, \tag{23}$$

where the offered loads are

$$\rho(i_1) = \frac{\lambda_0 + \lambda_h(i_1)}{\mu + \eta}, \qquad \rho(i_2) = \frac{\lambda_0 + \lambda_h(i_2)}{\mu + \eta}. \tag{24}$$

Proof. The proof breaks down into two cases: (a) no guard channels and (b) the presence of guard channels. Where there are no guard channels, the proof follows essentially from the fact that in general the ratio of successive states in the same iteration results in

$$\frac{P_{j+1}(k)}{P_j(k)} = \frac{(\rho(k)^{j+1}/(j+1)!)P_0(k)}{(\rho(k)^j/j!)P_0(k)} = \frac{\rho(k)}{j+1}. \tag{25}$$

If we have $\rho(i_1) > \rho(i_2)$, then we can move from

$$\frac{\rho(i_1)}{j+1} > \frac{\rho(i_2)}{j+1} \quad \text{back to} \quad \frac{P_{j+1}(i_1)}{P_j(i_1)} > \frac{P_{j+1}(i_2)}{P_j(i_2)}. \tag{26}$$

Similarly, if we start with the inequality between ratios of successive states in the same iteration, then we can end up with the inequality between loads $\rho(i_1)$ and $\rho(i_2)$. Hence the lemma holds in both directions where there are no guard channels.

In the presence of guard channels, there is an extra step involved in computing some of the ratios. Assume there are n nonguard channels. For $P_{j+1}(k)/P_j(k)$, where $0 \le j < n$, the situation is identical to the one where there are no guard channels. For $j = n$, however, the numerator represents a guard channel state, whereas the denominator is a nonguard channel state. For $n < j < C$, both the numerator and denominator are guard channel states. We show that these ratios in fact lead to the same expression, which in turn verifies the lemma.

Where $j = n$, we get

$$\frac{P_{j+1}(k)}{P_j(k)} = \frac{(\rho(k)^n \rho_h(k)/(n+1)!)P_0(k)}{(\rho(k)^n/n!)P_0(k)} = \frac{\rho_h(k)}{n+1} = \frac{\rho_h(k)}{j+1}. \tag{27}$$

Similarly, for $n < j < C$, we get

$$\frac{P_{j+1}(k)}{P_j(k)} = \frac{(\rho(k)^n \rho_h(k)^{j+1-n}/(j+1)!)P_0(k)}{(\rho(k)^n \rho_h(k)^{j-n}/j!)P_0(k)} = \frac{\rho_h(k)}{j+1}. \tag{28}$$

If we have $\rho_h(i_1) > \rho_h(i_2)$, then $\lambda_h(i_1) > \lambda_h(i_2)$. We can add the new call arrival rate (λ_0) to each side and then divide by $\mu + \eta$, giving us $\rho(i_1) > \rho(i_2)$. We can then move from

$$\frac{\rho(i_1)}{j+1} > \frac{\rho(i_2)}{j+1} \quad \text{back to} \quad \frac{P_{j+1}(i_1)}{P_j(i_1)} > \frac{P_{j+1}(i_2)}{P_j(i_2)}. \tag{29}$$

Similarly, if we start with the inequality between ratios of successive states in the same iteration, then we can end up with the inequality between loads $\rho_h(i_1)$ and $\rho_h(i_2)$. Hence the lemma also holds in both directions in the presence of guard channels. □

We use these lemmas for showing convergence in our approach. In the next subsection we state and prove these theorems.

3.1. Convergence theorems

The technique of showing that the successive iterations of $\lambda_h(k)$ produce calculated values for the handoff arrival rate that monotonically approach the actual value λ_h^* demonstrates that (11) always converges.

If the initial value $\lambda_h(0)$ equals λ_h^*, we must have $\lambda_h(1) = \lambda_h^*$, and the computation terminates. This is because in this case $P_j(0), 0 \le j \le C$, are the steady state probabilities of the Markov chain. Since the steady state distribution is unique, any initial value $\lambda_h(0)$ not equal to λ_h^* would yield a $\lambda_h(1)$ that is not equal to $\lambda_h(0)$. Hence we must have the following:

$$\lambda_h(1) \neq \lambda_h^* \implies \text{either } \lambda_h(1) > \lambda_h(0) \text{ or } \lambda_h(1) < \lambda_h(0). \tag{30}$$

Here are the theorems on monotonic convergence of the proposed algorithm.

Theorem 1. *Assume the use of* (11) *for the calculation of the successive values of $\lambda_h(k)$. If the initial value chosen for $\lambda_h(0)$ is not equal to λ_h^*, the sequence $\lambda_h(k)$, $k = 1, 2, \ldots$, is monotonic.*

Proof. In view of inequalities (30) we need to consider two cases: (1) $\lambda_h(1) > \lambda_h(0)$ and (2) $\lambda_h(1) < \lambda_h(0)$.

Case 1. In this case we inductively establish that if for some $m > 0$, $\lambda_h(m+1) > \lambda_h(m)$, then we must have $\lambda_h(m+2) > \lambda_h(m+1)$.

If $\lambda_h(m+1) > \lambda_h(m)$, by definition (see (1)) we have $\rho(m+1) > \rho(m)$. Therefore, by Lemma 3 we have

$$\frac{P_{j+1}(m+1)}{P_j(m+1)} > \frac{P_{j+1}(m)}{P_j(m)} \quad \forall\, j, 0 \le j < C. \tag{31}$$

Now, by Lemma 2,

$$\sum_{j=0}^{C} jP_j(m+1) > \sum_{j=0}^{C} jP_j(m). \tag{32}$$

Equation (11) now implies $\lambda_h(m+2) > \lambda_h(m+1)$.

Case 2. In this case we inductively establish that if for some $m > 0$, $\lambda_h(m+1) < \lambda_h(m)$, then we must have $\lambda_h(m+2) < \lambda_h(m+1)$.

If $\lambda_h(m+1) < \lambda_h(m)$, by definition (see (1)) we have $\rho(m+1) < \rho(m)$. Therefore, by Lemma 3 we have

$$\frac{P_{j+1}(m+1)}{P_j(m+1)} < \frac{P_{j+1}(m)}{P_j(m)} \quad \forall\, j,\ 0 \le j < C. \tag{33}$$

Now, by Lemma 2, we have

$$\sum_{j=0}^{C} jP_j(m+1) < \sum_{j=0}^{C} jP_j(m). \tag{34}$$

Equation (11) now implies $\lambda_h(m+2) < \lambda_h(m+1)$. □

The two cases considered above in the proof of Theorem 1 immediately, leads to the following two corollaries.

Corollary 1. *Assume the use of* (11) *for the calculation of the successive values of $\lambda_h(k)$. If the initial value chosen for $\lambda_h(0)$ is less than the actual value λ_h^*, the sequence $\lambda_h(k)$, $k = 1, 2, \ldots$, is monotonically increasing.*

Corollary 2. *Assume the use of* (11) *for the calculation of the successive values of $\lambda_h(k)$. If the initial value chosen for $\lambda_h(0)$ is greater than the actual value λ_h^*, the sequence $\lambda_h(k)$, $k = 1, 2, \ldots$, is monotonically decreasing.*

The following theorem asserts the convergence of the computation, in all cases, to the desired value.

Theorem 2. *Assume the use of* (11) *for the calculation of the successive values of $\lambda_h(k)$. For any initial value of $\lambda_h(0)$,*

$$\lambda_h^* = \lim_{k \to \infty} \lambda_h(k), \tag{35}$$

where $\{P_j(k)\}$, $0 \le j \le C$, is the steady state probability distribution of the one-dimensional finite, irreducible, positive recurrent Markov chain on $\{0, 1, \ldots, C\}$ determined by the parameters at the kth iteration of the algorithm.

Proof. If $\lambda_h(0) = \lambda_h^*$, then as remarked earlier the computation terminates and the result is true. If $\lambda_h(0) \neq \lambda_h^*$, by Theorem 1, $\lambda_h(k)$ is a monotone sequence. By Lemma 1,

$\sum_{j=l}^{C} P_j(k)$ is a monotone sequence in k for each l. Therefore all these sequences have a limit as $k \to \infty$, and consequently $P_j(k)$ has a limit for all j. Therefore, taking limits in (11), we get

$$\lim_{k\to\infty} \lambda_h(k) = \eta \sum_{j=0}^{C} j \lim_{k\to\infty} P_j(k). \tag{36}$$

Since the limits satisfy the balance equations, by uniqueness of the steady state distribution we must have $\lim_{k\to\infty} \lambda_h(k) = \lambda_h^*$. □

4. FASTER CONVERGENCE BY A BISECTION ALGORITHM

Although the monotonic algorithm given in Section 2.5 does converge, the rate is much slower than necessary for practical applications in cellular networks. A faster approach makes use of the fact that the successive values of $\lambda_h(k)$ are monotonic. The basic idea is to take two values, $low\lambda_h$ and $hi\lambda_h$, that are known to be lower and higher, respectively, than λ_h^*. These two values are averaged, and the result is deemed to be the testValue for λ_h. The testValue is then fed into the iterative process (11), which produces a resultValue.

By virtue of monotonicity, if the resultValue is less than the testValue, then we know that λ_h^* is less than the resultValue. In other words,

$$\begin{aligned} &\text{lowerValue} < \lambda_h^* < \text{resultValue}, \\ &\text{resultValue} < \text{testValue} < \text{higherValue}. \end{aligned} \tag{37}$$

In that case, we keep the same lowerValue and we make the resultValue the new higherValue. In the same manner, if a resultValue is greater than the testValue that produced it, then we know that λ_h^* is greater than the resultValue. Now the relationships are

$$\begin{aligned} &\text{lowerValue} < \text{testValue} < \text{resultValue}, \\ &\text{resultValue} < \lambda_h^* < \text{higherValue}. \end{aligned} \tag{38}$$

Here, the higherValue would remain the same, and the resultValue becomes the new lowerValue. The lower and higher values are averaged, which produces a new testValue. This continues until the difference between the lower and higher values is within the desired accuracy of the user.

For original lower and higher values, we use the lower and higher bounds for λ_h^*, namely, 0 and ηC. The foregoing is captured in Algorithm 3.

This approach is an improvement over the monotonic algorithm, which merely used the result from one iteration as the initial value for the next iteration. By taking advantage of the knowledge given to us by Corollaries 1 and 2, we know from the relationship between testValue and resultValue whether the actual value λ_h^* is greater than or less than the resultValue, and we can adjust the lower or higher bound accordingly as we hone in on the actual value. In fact, our approach is even stronger than a pure bisection, because we are able to use resultValue (and not just the testValue) as the new lower or higher value for the following iteration. Hence, the range [lowerValue, higherValue] shrinks by *more* than one-half with each iteration.

```
some small value ε > 0
low λ_h := 0
hi λ_h := ηC
while (hi λ_h − low λ_h > ε)
    Step 1: testValue := (low λ_h + hi λ_h)/2
    Step 2: update values for the offered load ρ and handoff
      load ρ_h per (1)
    Step 3: update the value of P_0 per (3)
    Step 4: update the values of state probabilities P_1
      through P_C per (2)
    Step 5: compute the new value for λ_h, that is,
      resultValue, per (11)
    Step 6: if (resultValue < testValue) then
              hi λ_h := resultValue
            else {resultValue > testValue}
              low λ_h := resultValue
endwhile
λ_h := (low λ_h + hi λ_h)/2
```

Algorithm 3: Bisection algorithm.

We illustrate with two charts the speed with which the proposed bisection algorithm can achieve a very accurate approximation of λ_h^* quickly. In Figure 6, a value of 0.1 was used for λ_0, and the result from the bisection algorithm is combined with results from Figures 1 and 4. Figure 7 is similar, using a value of 0.12 for λ_0 and combining with the results from Figures 2 (which did not converge) and 5 (which did converge, albeit somewhat slowly).

The convergence properties of the bisection algorithm can be expressed in a theorem.

Theorem 3. *The bisection algorithm converges. Moreover, for a given degree of accuracy $\epsilon > 0$, the number of iterations required to achieve that level of accuracy is on the order of*

$$\log_2 \frac{\eta C}{\epsilon}. \tag{39}$$

Proof. We begin with the maximum possible range of values for λ_h^*, which is $[0, \eta C]$. Because of Corollaries 1 and 2, with each iteration one end of the range is adjusted in the direction of the actual value of λ_h^*, always keeping the actual value within the range, and hence the range continues to shrink as the number of iterations increases. We note that the initial gap is simply $\eta C - 0 = \eta C$. The gap is actually divided by more than a factor of 2 with each iteration. That can be observed from the fact that testValue is the average of the current range endpoints, but resultValue replaces one of the endpoints for the next iteration (leaving the other endpoint

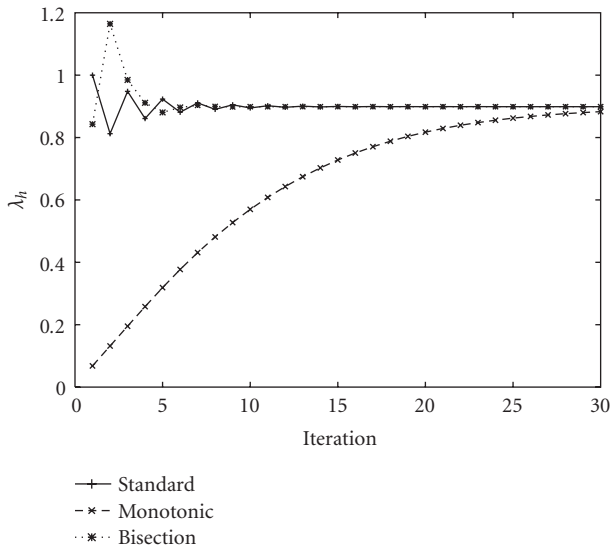

FIGURE 6: Comparison: $\lambda_0 = 0.1$.

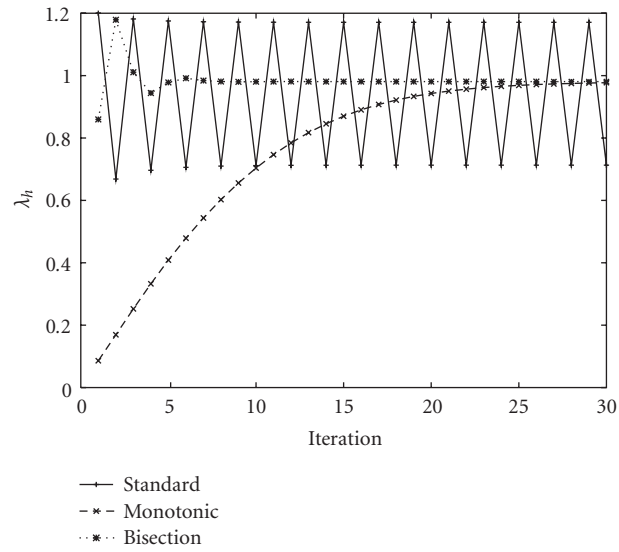

FIGURE 7: Comparison: $\lambda_0 = 0.12$.

intact). Hence, the gap for the next iteration is

$$\begin{aligned} &|\text{resultValue} - \text{other endpoint}| \\ &\quad < |\text{testValue} - \text{other endpoint}| \\ &\quad = \frac{\text{previous gap}}{2}. \end{aligned} \tag{40}$$

Now the logarithmic convergence can be established from the following classical argument. For a given value of $\epsilon > 0$, we need to keep dividing the gap until the range is within the desired degree of accuracy. The number of steps, call it m, needed to accomplish this can be expressed as

$$\frac{\eta C}{2^m} \leq \epsilon \Longrightarrow \frac{\eta C}{\epsilon} \leq 2^m, \tag{41}$$

which means

$$\log_2 \frac{\eta C}{\epsilon} \leq m. \tag{42}$$

The smallest integer n that satisfies this inequality is the maximum number of required iterations. This completes the proof of the theorem. □

5. MODEL VALIDATION

For validation of the accuracy of handoff call arrival rates obtained from the algorithms presented in previous sections, we developed a simulation model. The cell layout for our simulation model is shown in Figure 8. The 49 white cells are part of the model and the shaded ones show the wraparound neighbors. The wraparound topology is used, since it eliminates the boundary effect keeping exactly six neighbors for each cell [9].

We assume a static channel allocation scheme for cells, that is, the number of channels allocated to a cell does not change during the simulation. For the results reported here, all cells were assigned 20 channels. Mobility of terminals is modeled using a simple random walk, that is, a terminal moves to any of the current cell's neighbors with equal probability—1/6 for the hexagonal layout. New call arrivals into the network follow the Poisson distribution with mean λ calls/s. The call holding time and the cell dwell time follow exponential distributions with respective means $1/\mu$ and $1/\eta$ seconds. For obtaining good estimates of the parameters, each simulation study was run for 1 000 000 new calls. Note that the assumptions for the simulations are identical to those to our analysis. Our extensive studies have shown a close match between the theoretical and simulation results.

Some typical results are shown in Table 1. We varied call arrival rate from 0.06 to 0.2 calls per second. The average call holding time and cell dwell time were kept constant at 120 seconds and 12 seconds, respectively. Out of the 20 channels, 4 were used as guard channels. As can be seen from the table, handoff call arrival rates calculated by the algorithm and obtained from simulations agree up through the hundredth place. Therefore, handoff call arrival rates calculated from the presented algorithm are very accurate.

6. CONCLUSION

Since the late eighties, the modeling and analysis of the performance of wireless networks have produced sets of nonlinear equations with interrelated parameters. These nonlinear equations have no closed-form solution, so the numerical values of the parameters are calculated by iterative algorithms. When these iterations fail to converge, however, the precise values of the parameters cannot be determined.

Using a Markov chain to model a wireless cellular network, we discussed a common expression for calculating the handoff arrival rate iteratively. We then provided for illustration an instance where the sequence of iterative values fails

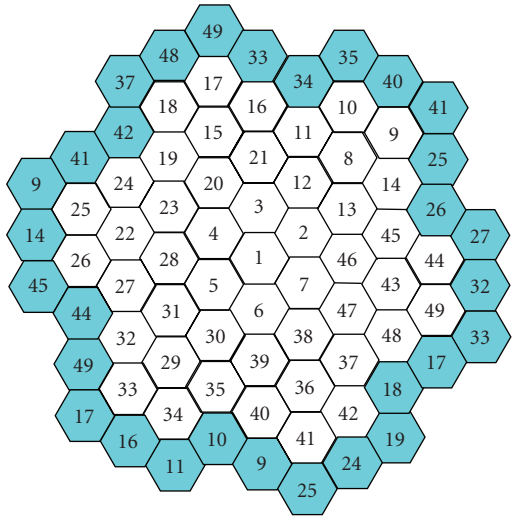

Figure 8: Cell layout for the simulation model.

Table 1: Comparison of theoretical and simulation results.

New call	Handoff call arrival rates	
arrival rates	Theoretical	Simulation
0.06	0.598 098	0.597 954
0.07	0.691 469	0.690 925
0.08	0.774 357	0.773 294
0.09	0.843 334	0.842 125
0.1	0.899 022	0.898 609
0.11	0.944 041	0.944 403
0.12	0.980 989	0.982 910
0.13	1.011 970	1.012 820
0.14	1.038 360	1.041 210
0.15	1.061 210	1.063 370
0.16	1.081 260	1.084 360
0.17	1.099 070	1.102 840
0.18	1.115 020	1.117 960
0.19	1.129 430	1.133 820
0.2	1.142 550	1.145 900

to converge. After explaining the reason for the nonconverging oscillations, we gave an alternate simple iterative algorithm that generates a monotonic sequence and proved that the monotonic sequence always converges. Lastly, we refined this algorithm and, drawing upon the earlier results of this paper, set forth another algorithm that converges logarithmically.

The proposed algorithm can be used in existing cellular network optimization and call control algorithms [10, 13].

ACKNOWLEDGMENT

We would like to thank the reviewers for their constructive comments on an earlier version of the paper. The current version has greatly benefited from those comments.

REFERENCES

[1] Y. Fang, I. Chlamtac, and Y.-B. Lin, "Modeling PCS networks under general call holding time and cell residence time distributions," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 893–906, 1997.

[2] Y.-B. Lin and I. Chlamtac, "Effects of Erlang call holding times on PCS call completion," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 3, pp. 815–823, 1999.

[3] S. Mopati, "Dynamic adjustment of call pre-blocking parameter using fuzzy associative memory," M. S. thesis, University of Miami, Coral Gables, Fla, USA, 2002.

[4] S. Mopati and D. Sarkar, "Call admission control in mobile cellular systems using fuzzy associative memory," in *Proc. 12th International Conference on Computer Communications and Networks (ICCCN '03)*, pp. 95–100, Dallas, Tex, USA, October 2003.

[5] R. N. S. Chandra, "FAM-based CAC algorithms for interference limited mobile cellular CDMA systems," M. S. thesis, University of Miami, Coral Gables, Fla, USA, 2003.

[6] R. N. S. Chandra and D. Sarkar, "Call admission control in mobile cellular CDMA systems using fuzzy associative memory," in *Proc. IEEE International Conference on Communications (ICC '04)*, vol. 7, pp. 4082–4086, Paris, France, June 2004.

[7] D. Fedyanin and D. Sarkar, "Iterative algorithms for performance evaluation of wireless networks with guard channels," *International Journal of Wireless Information Networks*, vol. 8, no. 4, pp. 239–245, 2001.

[8] Y.-B. Lin, S. Mohan, and A. Noerpel, "Queueing priority channel assignment strategies for PCS hand-off and initial access," *IEEE Transactions on Vehicular Technology*, vol. 43, no. 3, part 2, pp. 704–712, 1994.

[9] L. Ortigoza-Guerrero and A. H. Aghvami, "A prioritized handoff dynamic channel allocation strategy for PCS," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 4, pp. 1203–1215, 1999.

[10] R. Ramjee, D. Towsley, and R. Nagarajan, "On optimal call admission control in cellular networks," *Wireless Networks*, vol. 3, no. 1, pp. 29–41, 1997.

[11] J. Hou and Y. Fang, "Mobility-based call admission control schemes for wireless mobile networks," *Wireless Communications and Mobile Computing*, vol. 1, no. 3, pp. 269–282, 2001.

[12] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall, Upper Saddle River, NJ, USA, 1996.

[13] G. Harine, R. Marie, R. Puigjaner, and K. Trivedi, "Loss formulas and their application to optimization for cellular networks," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 3, pp. 664–673, 2001.

[14] D. Sarkar and T. Jewell, "Convergence in the calculation of the handoff arrival rate: A log-time iterative algorithm," Tech. Rep. CS-TR-SJ-01, University of Miami, Coral Gables, Fla, USA, August 2002.

[15] E. L. Lehmann, *Testing Statistical Hypotheses*, John Wiley & Sons, New York, NY, USA, 1959.

[16] J. G. Shanthikumar, "Stochastic majorization of random variables with proportional equilibrium rates," *Advances in Applied Probability*, vol. 19, pp. 854–872, 1987.

Dilip Sarkar received the B.Tech. (honors) degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in May 1983, the M.S. degree in computer science from the Indian Institute of Science, Bangalore, India, in December 1984, and the Ph.D. degree in computer science from the University of Central Florida, Orlando, in May 1988. From January 1985 to August 1986, he was a Ph.D. student at Washington State University, Pullman. He is currently an Associate Professor of computer science at the University of Miami, Coral Gables. His research interests include parallel and distributed processing, the web computing and middleware, multimedia communication over broadband and wireless networks, fuzzy systems, neural networks, and concurrent multipath transport protocols. In these areas, he has guided several theses and has authored numerous papers. He is a Senior Member of the IEEE, a Member of IEEE Communications Society. He has served on the Technical Program Committees of numerous conferences including Globecom, ICC, and INFOCOM. He was a recipient of the Fourteenth All India Design Competition Award in electronics in 1982.

Theodore Jewell received the A.B. degree from Harvard College in 1975. During the academic year 2000–2001, he attended the University of Miami. While there and in the summer of 2002, he was a student of and worked with Dr. Dilip Sarkar. He received the M.S. degree in computer science from Yale University in 2003. He is currently a member of the faculty at The Taft School, Watertown, Conn, where he teaches courses in mathematics and computer science.

S. Ramakrishnan received his B.Stat. (honors), M.Stat., and Ph.D. degrees in 1974, 1975, and 1982, respectively, from the Indian Statistical Institute, Calcutta, India. Since then he has been a Faculty Member at the University of Miami where he is currently an Associate Professor in the Department of Mathematics. His research interests include the foundations of probability theory and stochastic processes, theory of gambling, and more recently, discrete-time queueing networks, and theoretical computer science.