User Agent Migration Policies in Wireless Networks

Ramachandran Ramjee & Thomas La Porta
Bell Labs, Lucent Technologies
Holmdel, NJ
ramjee,tlp@bell-labs.com

Jim Kurose & Don Towsley
Dept of Comp. Science
Univ of Massachusetts
kurose,towsley@cs.umass.edu

Abstract

Wireless networks often employ network-based user agents as proxies for mobile users. In this paper, we consider the fundamental problem of designing migration policies for these user agents. We first introduce a general framework for analyzing user agent migration policies and then highlight, through analysis and simulation, the numerous parameters and tradeoffs that dictate the design of migration policies. We evaluate these policies in the context of both homogeneous and heterogeneous networks, and in the presence and absence of processing overheads due to migration. Finally, we identify two simple threshold-based policies that deliver very good performance over a wide range of system parameters and configurations. To our knowledge, this is the first paper to propose and evaluate policies for migration of user agents.

1 Introduction

Wireless networks are characterized by significant differences in communication and processing capabilities between the wireless endpoints and fixed servers. One common technique to overcome the differences in these networks is to introduce an intermediary server through which the wireless endpoint communicates with the fixed network. The intermediary server, which resides in the fixed network, has access to high bandwidth links and large computational resources. These resources can be utilized by moving selected functionality from the mobile terminal to this intermediary server. When this intermediary server performs actions on behalf of a mobile user, it is termed a "user agent". The need for such an agent inside the fixed network, acting on behalf of a mobile user, has been proposed by several researchers [1, 2, 4, 8].

User agents may operate on control traffic such as signaling messages, or data traffic to perform functions such as audio and video transcoding. As wireless applications increase in complexity from simple voice services to advanced multiparty multimedia services, user agents themselves become increasingly complex and consume significant processing and bandwidth resources. Thus, allocation of user agents to processors and the impact of user mobility on agents becomes an important issue. However, while agents themselves have received considerable attention, the issue of user agent mobility has received less attention to date. In many applications, this issue has been resolved somewhat in an ad-hoc manner, based on the environment in which the application operates. For example, user agents in [4] are placed at fixed well known locations (such as the mobile user's home workstation) since the application operates in a campus area environment. Similarly, there is the concept of a home agent in the Mobile

IP standard [7]. On the other hand, user agents in [8] are migrated along with the mobile user since the application operates over a wide-area environment.

When the user agents are fixed, an appropriate initial placement of agents can help to balance the computational load that they place on the network servers. For example, load balancing can be achieved by distributing the user agents evenly among all the network servers and keeping them fixed. However, such an approach could result in large distances separating mobile users from their agents which could translate to inefficient bandwidth usage and slower responses to mobile user queries due to network congestion (control messages and, in some cases, application data traffic to/from the mobile user must flow through the agent). When user agents are always migrated to locations closest to the mobile users, network bandwidth can be utilized more efficiently and mobile user queries may receive faster responses. However, migrating the agent every time the user moves can result in severe load imbalances among the fixed network servers on which these agents execute, particularly when there are "hot spots" in the network.

These two solutions represent two ends of a spectrum of policies: one corresponding to a perfectly load balanced system with potentially large distances separating mobile users from their agents and the other corresponding to a highly imbalanced system with no separation between the mobile users and their agents. Our objective in this paper is to study the dynamics of mobile user agents and propose suitable policies for migrating these agents in order to balance the load among the network servers, while at the same time maintaining small distances between mobile users and their agents.

This paper makes three important contributions. First, we introduce a general framework for examining user agent migration policies in wireless networks. Second, through analysis and simulation, we illustrate the numerous trade-offs and parameters that dictate the design of migration policies. Finally, we identify two threshold based migration polices: a *Count* policy that limits the maximum number of agents on a given processor and a *Distance* policy that gives preference to migration of agents that are farther away from their users. We show that the Count policy is preferred when it is beneficial for the wireless network to operate with a large number of migrations (e.g., because of a high penalty associated with distance separating users and their agents or low overhead in migration) and the Distance policy is preferred when it is beneficial for the wireless network to operate only with a small number of migrations (e.g., because of very low distance penalty or significant overhead in migration). We further show that both the Count and Distance policies deliver very good performance over a wide range of system configurations and parameters.

The remainder of this paper is organized as follows. In the next section, present an overview of our network architecture, highlighting the operations of user agents and identifying several user agent migration policies. In Section 3, we present our system model with the associated parameters, notation, and the underlying assumptions. In Section 4, we examine two baseline policies - a policy that keeps the user agent at a fixed location and another that always migrates the user agent along with the mobile user. The performance results obtained in this section motivate the examination of more sophisticated migration policies, Count and Distance, which is the focus of Section 5. In

Section 6, we compare the performance of several migration policies. In Section 7, we show how heterogeneity in the rate and direction of user movements can be incorporated into our framework. In Section 8, we study the impact of processing overhead due to user agent migration. Finally, we present our conclusions in Section 9.

2 Network Architecture and Migration Policies

In this section, we present the network architecture and discuss the various migration policies considered in this work.

2.1 Network Architecture

Typically, wireless networks are partitioned into regions called cells. In each cell, a base-station manages the allocation of wireless communication resources or channels to the mobile user. Every mobile user is represented by a network-based user agent which performs processing on behalf of the mobile user. These user agents execute on processors co-located in the base-stations inside the wireless access network infrastructure. These processors are assumed to be fully dedicated for user agent processing on behalf of the mobile users.

When a mobile user issues a request, the request is processed by the user agent representing that mobile user. For example, a request may be to set up a multiparty multimedia call in an advanced telecommunication application. In this case, processing this request by the user agent at the network server involves, among other things, decoding/expanding the request, determining the appropriate services (e.g., special billing) to be invoked on behalf of the mobile user, performing any negotiations with the other users (or their agents), and contacting the appropriate network entities to establish the multimedia connection. In applications such as video transcoding, the data flowing to the mobile user will have to be processed by the user agent. In this case, processing data packets corresponds to processing requests at the user agent.

In this paper, we consider the migration of user agents among the processors in the wireless access infrastructure. In this context, security is a very important issue. Security is one of the motivations behind our considerations of migration of state information rather than actual user code. For example, in [8], user agent migrations require the transfer of only about 100 bytes of state information. This transfer should not cause security concerns if the migration is within a wireless service provider's access network since it is a single administrative domain. If the migration is between two administrative domains, an arrangement similar to the one used in current cellular networks for downloading profiles between Home Location Registers and Visitor Location Registers is necessary. Also, if the migration has to traverse through firewalls, the state information can be transferred through appropriate tunneling mechanisms. Recent work, such as the Media Gateway Architecture [2] which provides transcoding and other active services for fixed clients, demonstrates that even uploading user code in the form of application level "servents" by fixed clients into network servers is a very useful and feasible service. They propose to use type-safe languages such as SafeTCL or Java for these applications. Our work can be applied in these frameworks as well.

Given the various considerations, the user agent may or may not be executing at the processor in the cell in which the user is currently located. If the user agent is not executing in the cell in which the mobile user is currently

located, then the request (or data) must be forwarded over the fixed network to the user agent. This leads to additional bandwidth usage and delays in processing the request.

2.2 Migration Policies

A migration policy makes decisions about migrating user agents among the processors in the base-stations. Migration policies can be classified as either *active* or *reactive*. A reactive migration policy makes decisions about migrating a user agent only when the associated mobile user moves from one cell to another. An active migration policy may migrate a user agent even when the associated mobile user remains in its current cell. Migration policies can also be classified as *current location* and *neighborhood location* policies based on the type of decision they make. A current location policy makes a simple yes/no decision about whether to migrate the user agent to the processor attached to the cell in which the mobile user is currently located. A neighborhood location policy would make a decision about whether to migrate the user agent to any one of the processors attached to cells in the "neighborhood" of the cell in which the mobile user is currently located. The number of cells that comprise a neighborhood could itself be a parameter of the neighborhood location policy.

In this paper, we will study the performance of various current location, reactive type migration policies. We evaluate the migration policies using two performance metrics: the *average response time* for processing user requests and the *average distance* separating mobile users from their agents. The goal of these migration policies is to keep the values of these two metrics as low as possible. Note that the former metric reflects the extent to which the overall processing load in the network is balanced among the network processors; if the load is highly imbalanced, the average response times will also be high. Thus, keeping the average response time low requires keeping the load balanced. The second metric, the average distance separating mobile users from their agents, reflects a measure of the amount of bandwidth needed in the wired network, to support communication among the mobile users and their agents. In order to reduce bandwidth requirements, a migration should thus also attempt to locate the user agent "close" to the mobile user.

We will now describe two simple migration policies, the Always-migrate and the Never-migrate policies.

- **Always-migrate policy:** As the name suggests, the Always-migrate policy migrates the user agent along with the mobile user each time the mobile user moves from one base-station to another. This migration is accomplished by transferring the state of the user agent from the network server in the mobile user's old cell to the network server in the mobile user's current cell. This policy avoids the distance penalty associated with propagating messages/data between mobile users and their agents. However, the Always-migrate policy disregards any load imbalances that arise when a large number of users congregate in a cell.
- **Never-migrate policy:** The Never-migrate policy assigns the user agent to a fixed processor and, as the name suggests, never migrates the agent. The assignment can be performed, for example, using a well-known algorithm so that requests from mobile users are forwarded to the appropriate processor in which their respective user agents execute. We assume that the user agent can be assigned to any of the processors in the wireless

network at the discretion of the policy. Under this assumption, the Never-migrate policy can evenly divide the user agents among the network servers, thereby balancing the processing load in the servers. However, as the mobile users move, the messages/data between the mobile users and their agents may have to flow long distances. This may lead to inefficient use of wired network bandwidth.

Note that anchoring the user agent in the Never-migrate policy leads to triangular routing, i.e., when a correspondent host C is communicating with mobile user M, the packets always go through user agent A before reaching M. Routing in the Always-migrate and Never-migrate policies can be represented in the forms ($C \leftrightarrow M$) and ($C \leftrightarrow A \leftrightarrow M$) respectively. Now consider the overall routing efficiency of both policies. Assuming that C can be anywhere in the network with uniform probability, for every C such that the path $C \leftrightarrow M$ has x more hops than the path $C \leftrightarrow A$, there exists another C such that the path $C \leftrightarrow M$ has x fewer hops than the path $C \leftrightarrow A$. Thus, these terms cancel each other when we aggregate over every possible C in the network while the path $A \leftrightarrow M$ remains in the case of the Never-migrate policy. The average value for $A \leftrightarrow M$ is the highest for the never-migrate policy and zero for always-migrate policy. Thus, these policies represent two extremes in the case of distance separating the user and the agent in a spectrum of policies. Similarly, if one considers processing load as a measure, Always-migrate and Never-migrate again represent two extremes: in this case, Never-migrate is perfectly load balanced while Always-migrate completely disregards the load in the network processors.

One of the main goals of this work is to identify suitable migration policies that operate between the two ends of this spectrum, thereby balancing the processing delays due to load imbalances with the average distance separating mobile users and their agents. In this paper, we examine two such migration policies in detail[1]. The policies are fairly intuitive and were chosen based on the simplicity of their implementations. In more general terms, these policies can be classified as threshold type policies since policy decisions are made based on simple threshold parameters. In addition to their simplicity, we chose to examine threshold type policies based on distributed system load balancing work in [3] that showed that "state information beyond that used by Threshold, or a more complex usage of state information, is of little benefit". Since user agent migration policies share several similarities with the load balancing policies, threshold-based migration policies offer a good first choice.

The two additional policies we will consider are:

- **Count policy:** In this policy, an upper limit (threshold, T) is imposed on the maximum number of user agents that can be present in a processor at any given time. User agents migrate to the destination cell along with the mobile user only if the number of user agents at the destination processor is less than T. Otherwise, the user agent remains at its current location until the next move by the mobile user (at which point another attempt is made to transfer the agent). If the agent and the user are not co-located, the messages from (to) the mobile user to (from) the user agent are routed over the fixed network, leading to additional bandwidth usage.

[1] A discussion of three other variations of these two migration policies that we considered can be found in [9].

- **Distance policy:** This policy makes migrations decisions based on whether a processor is underloaded or not. We will give a precise definition of what is meant by an underloaded processor in the next section. For now, we informally define an underutilized processor to be one having lower than expected load. Under this policy, all user agent migrations are permitted if the server is underloaded. Otherwise, only those user agents that are at a distance greater than T hops away from their user are migrated (into the cell to which the mobile user has just moved).

The Count policy works by restricting the maximum number of agents that can be present in the server at any given time, thereby restricting large load imbalances from occurring. Under this policy, while the maximum load at any processor is controlled there may be cases where some mobile users are located very far away from their agents. On the other hand, under the Distance policy, while the maximum distance separating mobile users from their agents is controlled, there may be occasional severe processing overloads. These two policies, while operating in the middle of the spectrum of migration policies, control the average response time and average distance metrics respectively. As we shall see later, these policies deliver significant performance improvements over the Always-migrate and the Never-migrate policies.

3 System Model

We model the wireless network as a homogeneous network with a fixed user population, U, and a fixed number of homogeneous cells, labeled $k = 0, 1, \ldots, C-1$. We will relax this homogeneity assumption to model cell heterogeneity in a later section. The mobility behavior of the user is modeled by a two-dimensional random walk model. The mobile user stays in a cell for a period of time which is exponentially distributed with mean $1/\lambda$. The mobile user then moves to one of the neighboring cells with equal probability. These are fairly common assumptions for modeling user mobility in wide-area wireless environments and have been used in [10, 11].

Each user is served by a network-based user agent as detailed in Section 2. Requests arrive at the user agent according to a Poisson process with rate λ_c. Request processing times are assumed to be exponentially distributed with mean $1/\mu$. We also assume that each user can have multiple pending requests, although this assumption can easily be relaxed. As the mobile user moves, the user agent may be migrated along with the mobile user depending on the migration policy used. For now, we assume that there is negligible processing overhead due to user agent migration. In a later section, we will study the behavior of the system in the presence of migration overheads.

An exact model of this system is inordinately complex. Instead we develop an approximate analysis based on the model of a single cell. This analysis is conjectured to be exact asymptotically as the number of neighbors goes to infinity [3]. Our simulation results, presented later, indicate that the approximate analytic results are very close to the simulation estimates for the parameter value ranges in which we are interested in the case of a wireless network comprising of hexagonal/octagonal cells with 6/8 neighbors.

Let us thus focus on an individual cell. Based on the user mobility model discussed earlier, the number of users in the cell is a random variable, $n(t), t > 0$. Under the assumption that the system exhibits steady state

behavior, define $n = \lim_{t\to\infty} n(t)$. The average number of users in a given cell is simply $E[n] = U/C = N$. The number of user *agents* in a given cell at time t is also a random variable, denoted $m(t), t > 0$. Note that this may differ from the number of users in the cell because some migration policies will not always result in a user and its agent being co-located in the same cell. However, the average number of user agents is $E[m] = M = N$ (where $m = \lim_{t\to\infty} m(t)$), since each user is represented by exactly one user agent. When a processor has less than N user agents, we define that processor as an *underloaded* processor.

We also place a limit on the number of users (L) that can be simultaneously supported in a single cell. This results from a cell's inability to support more than L users due to lack of wired/wireless bandwidth or other resources. In this paper, we have chosen L large enough (and greater than N) so that $P(n(t) = L)$ is extremely small. In other words, the system is assumed to be engineered with sufficient resources so that the probability of a user being denied service due to lack of wireless resources (as opposed to computational resources) is very small.

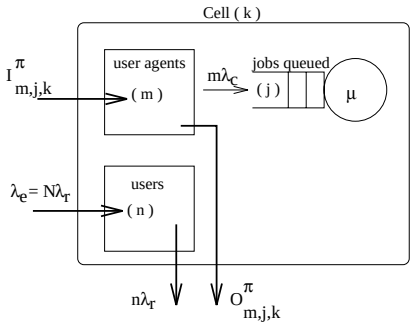


Figure 1: System Model and parameters

Figure 1 highlights the parameters and notations used in modeling the wireless network for a given cell k. A server with j user agent requests queued is shown. Note that the request arrival rate to the server, $m\lambda_c$, depends on the number of user agents, m, resident at that instant in the cell. The flow rate of *user agents* into and out of the cell is represented by $I^{\pi}_{m,j,k}$ and $O^{\pi}_{m,j,k}$ respectively, where π denotes the migration policy being used. The flow rate of *users* into and out of the cell are given by $\lambda_e = N\lambda_r$ and $n\lambda_r$ respectively.

Based on the network architecture described in the previous section and our modeling assumptions, a Markov model of the wireless network under various migration policies can be constructed. Given a migration policy π, we are interested in deriving two main performance measures using the model: R^{π}, the average response time for job requests at the depicted server and D^{π}, the average distance in hops, which separates mobile users from their agents. In the following section, we will discuss the models and their solutions of the Always-migrate and the Never-migrate policies. In the subsequent section, we will discuss the models of the Count and Distance policies.

4 Analysis of Baseline Migration Policies

In this section, we model the behavior of the Always-migrate and Never-migrate policies. Our goal is to derive the average response times and average distances under the two policies and determine the factors that influence the

performance of these policies.

4.1 Never-migrate Policy

Recall that under this policy, the U user agents are initially evenly distributed among the network processors so that all processors in the network contain N user agents each. The user agents in this policy then remain fixed while the mobile users may contact their agents from different locations.

Since the agents are fixed, the flow rates of agents into and out of a cell, $I_{\cdot,\cdot,\cdot}^{Never}$ and $O_{\cdot,\cdot,\cdot}^{Never}$ are both zero. The average response time under this policy is then simply the response time for a M/M/1 queue with an arrival rate of $N\lambda_c$ and a service rate of μ. Thus, the average response time, R^{Never}, is given by

$$R^{Never} = \frac{1}{\mu - N\lambda_c} \tag{1}$$

We now derive the average distance, D^{Never}, for a wrapped square mesh network topology with $C = c^2$ octagonal cells where $C > 8$. The calculation of D for other topologies and different number of neighbors will be similar. Since we are using the two dimensional Random walk mobility model in a homogeneous network, the average amount of time spent in each cell of the wireless network by a mobile user will be the same and equal to $1/\lambda_r$. Let d_i^γ denote the number of cells that are at a distance of i hops from the location of a given user agent, where γ denotes the number of neighbors to any given cell. It can be easily shown that

$$d_i^8 = \begin{cases} 1, & i = 0 \\ 8i, & 1 \leq i \leq \lfloor (c-1)/2 \rfloor \\ (4i-1), & \text{if} \quad c = 2i \end{cases}$$

The average distance, D^{Never}, is simply the summation of the term $d_i^8 i/C$ for different i (hop counts) since users spend the same average amount of time in each cell. Thus,

$$D^{Never} = \frac{\sum_{i=0}^{\lfloor c/2 \rfloor} i d_i^8}{C} \tag{2}$$

Simplifying equation (2), one can show that D^{Never} is directly proportional to the diameter of the wireless network. Thus, the average distance can be significant in large networks.

4.2 Always-migrate Policy

Under the Always-migrate policy, user agents are always located at the processors attached to the cell in which the mobile user is currently resident. Thus, this policy adopts the view that minimizing the distance of separation between mobile users and their agents is the most important metric. This may lead to load imbalances in the processors which will translate into higher average response times. We now proceed to derive the average response times for user agent requests in the Always-migrate policy.

A processor in a wireless cell under the Always-migrate policy can be modeled using a two dimensional Markov process with state (m, j) where m denotes the number of user agents and j the number of requests queued at the processor. The variable m also represents the number of mobile users present in the cell since the agents are always migrated along with the mobile user. In this system, the mean flow rate of user agents into a cell equals the mean flow rate of users, λ_e. Thus, $I_{\cdot,\cdot,\cdot}^{Always} = \lambda_e = N\lambda_r$, where λ_r is the flow rate of a single user. The mean flow rate of user agents out of a cell, $O_{m,\cdot,\cdot}^{Always} = m\lambda_r$, where m denotes the number of agents currently present in the cell. The arrival of a user agent (mobile user) from the neighboring cells results in a state transition to $(m+1, j)$, while the departure of a user agent (mobile user) results in a state transition to $(m-1, j)$.

The average distance separating the users and their agents, D^{Always}, is clearly 0 under this policy. We now outline our analytical approach for determining the average response time for user agent requests.

4.2.1 Analysis

We study the system analytically based on the matrix-geometric methods developed by Marcel Neuts [6]. This system can be interpreted as an M/M/1 queue in a random environment. The random environment arises here as a result of user's mobility. The environment is described by a Markov process with the following infinitesimal generator

$$Q = \begin{vmatrix} -\lambda_e & \lambda_e & 0 & 0 & 0 & \dots & 0 \\ \lambda_r & -\lambda_e - \lambda_r & \lambda_e & 0 & 0 & \dots & 0 \\ 0 & 2\lambda_r & -\lambda_e - 2\lambda_r & \lambda_e & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & L\lambda_r & -L\lambda_r \end{vmatrix}$$

The entries $q_{ij}, i \neq j$ in Q represent the rates of transitions from (i, k) to (j, k) in the two dimensional Markov process described earlier. The diagonal elements are the negatives of the rates at which departures occur from the various states.

Let ω be the stationary probability vector of Q. Then, ω is defined by

$$\omega Q = 0$$

$$\omega e = 1$$

where e denotes a unit column vector.

Solving the above equations, we find

$$\omega_j = \frac{N^j}{j! \Sigma_{j=0}^{L} \frac{N^j}{j!}}$$

The infinitesimal generator Matrix for the original system can be written as

$$G = \begin{vmatrix} Q-\Delta(\lambda) & \Delta(\lambda) & 0 & 0 & 0 & \dots \\ \Delta(\mu) & Q-\Delta(\lambda+\mu) & \Delta(\lambda) & 0 & 0 & \dots \\ 0 & \Delta(\mu) & Q-\Delta(\lambda+\mu) & \Delta(\lambda) & 0 & \dots \\ 0 & 0 & \Delta(\mu) & Q-\Delta(\lambda+\mu) & \Delta(\lambda) & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{vmatrix}$$

with $\Delta(\mu) = Diag(\mu, \mu, \dots, \mu)$ and $\Delta(\lambda) = Diag(0, \lambda_c, 2\lambda_c, \dots, L\lambda_c)$ where $Diag$ denotes the diagonal elements of a diagonal matrix of order $L+1$.

The mean number of jobs in the system, J, is then given by [6]:

$$J = \omega(I-S)S(I-S)^{-2}e \tag{3}$$

where S is a $L+1$ by $L+1$ matrix which is the solution of the following matrix quadratic equation:

$$S^2\Delta(\mu) + S[Q-\Delta(\lambda+\mu)] + \Delta(\lambda) = 0. \tag{4}$$

Equation (4) can be rewritten as

$$S = [S^2\Delta(\mu) + \Delta(\lambda)][\Delta(\mu+\lambda) - Q]^{-1}$$

This can be further expanded to

$$S(i) = [S^2(i-1)\Delta(\mu) + \Delta(\lambda)][\Delta(\mu+\lambda) - Q]^{-1}$$

To determine S, we start with $S(0) = 0$ and iterate by successive substitutions. It can be shown that [6] the sequence of matrices $S(i)$ obtained through successive iteration is nondecreasing and $S(i) \to S$.

Once we obtain the mean number of requests in the system, J, through equation (3), the average response time, R, can be calculated using Little's law. Thus,

$$R^{Always} = J/(N\lambda_c) \tag{5}$$

4.2.2 Simulation

In order to verify the accuracy of the decomposition approximation used in the analytical model, we developed a discrete event simulator to model the wireless network with user agents processing requests on behalf of mobile users. The simulator also serves as a basis for studying the behavior of a more complex wireless network (e.g., with migration overhead incorporated) which is difficult to study analytically.

To simulate a very large network, the authors in [10] advocate a wrap-around topology. This approach eliminates the boundary effects in an unwrapped topology. It also results in a faster simulation since we need fewer cells to

achieve high confidence intervals in the measured values. Thus, we simulated our network using a wrapped mesh topology with the number of cells ranging from 64 to 200. Initially, each of the cells contain N users and each user agent is initially co-located with its associated user. We generate arrival events of workload to the user agent according to a Poisson process with rate (λ_c). We also generate handoff events for each user such that the user spends an exponential amount of time with mean $1/\lambda_r$ in each cell. The movement of the mobile users is based on a two-dimensional random walk model. In this model, the mobile users move to one of their neighboring cells with equal probability. The simulator allows for 4, 6, and 8 neighbors per cell for modeling rectangular, hexagonal, and octagonal cells respectively. The movement of the mobile user from one cell to its neighbor results in the generation of the handoff event.

When a given mobile user initiates a handoff, a decision is made about whether the corresponding user agent should be migrated or not, according to the specific migration policy. We assume for now that the processing involved in the migration is negligible compared to job request processing, and thus the migration of the user agent is accomplished instantaneously. The impact of migration overhead costs is examined in section 8. The simulation is terminated such that the confidence interval width of the 95% confidence level of the response time was within 5% of the point value. At least 10,000,000 events were simulated in each run to ensure that steady state is attained.

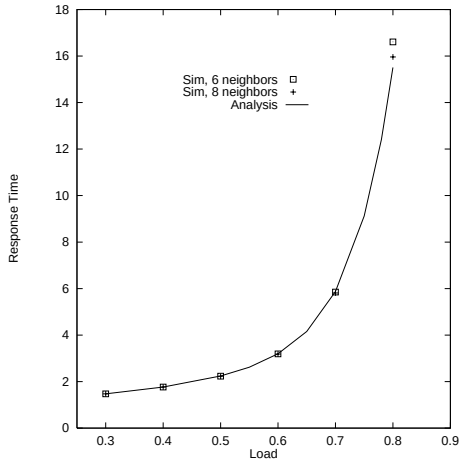


Figure 2: Always-migrate policy: Average Response Time vs Load through Simulation and Analysis. All quantities in this and other graphs are normalized to units of $1/\mu$.

The average load to a base-station processor, termed the system load, is given by $N\lambda_c/\mu$. Figure 2 plots the normalized average response time (obtained through both analysis using equation 5 and simulation) versus the system load when 20 users are initially present in each cell in a homogeneous network. Data points from simulations corresponding to the two cases of cells modeled as hexagonal and octagonal are shown. The only noticeable difference between the simulation results and the analysis occurs at high loads. Even when the average load is 0.8, the simulation results differ from analysis only by 3% in the case of octagonal cells. It is clear from the figure that the decomposition approach results in a very good approximation to the real system.

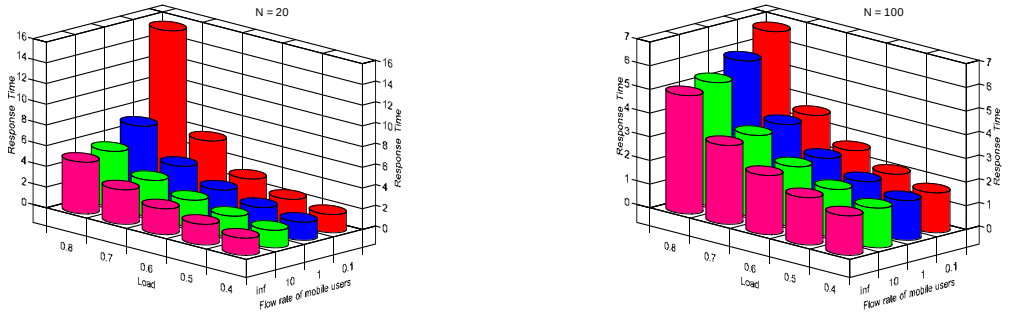


Figure 3: Average Response Times for $N = 20, 100$ for the Never-migrate (labeled inf) and the Always-migrate policies (with different mobility rates)

4.3 Results

We now examine the sensitivity of the average user response time to the mean user request rate λ_c, the mean request processing time $1/\mu$, the mean user flow rate λ_r and the average number of users per cell, N. The system load is given by $N\lambda_c/\mu$ while the net flow rate of mobile users both into and out of a cell is $N\lambda_r$.

To examine the sensitivity of these parameters to the average response time, we first choose a value of N and then compute the average response time for the Never-migrate and the Always-migrate policies. Since the user agents in the Never-migrate policy are fixed, R^{Never} is affected only by the system load. Thus, we compute R^{Never} using equation (1) for different values of system load. We compute R^{Always} using equation (5) for different values of system load and $N\lambda_r$. To examine the impact of N on the average response time, we perform the computations described above for different values of N.

In Figure 3, the z-axis plots the average response time for processing a user request in the homogeneous network. On the x-axis, the average load varies from 0.4 to 0.8. On the y-axis, the net flow rate has values 0.1,1.0,10.0 and inf. The series under inf corresponds to the average response times using the Never-migrate policy where the system is perfectly load balanced.

From Figure 3, we make two important observations. First, for a given average load, we observe that the average response time decreases as a function of the average number of users per cell, N, all other factors (including the overall load) being the same. For example, with a load of 0.8 and a flow rate of 0.1, we have an average response time of 15.5 for $N = 20$ as compared to 6.75 for $N = 100$ and 5.0 when the system is perfectly balanced.

This effect of N on the average response time can be explained by considering our mobility model. If we only consider the distribution of number of users per cell, the model degenerates into a M/M/L/L model, where L is the maximum number of users admissible in each cell. The probability distribution of number of users in each cell, P_i, is simply

$$P_i = \frac{\frac{N^i}{i!}}{\sum_{j=0}^{L} \frac{N^j}{j!}} \tag{6}$$

When N is small, the standard deviation of the number of users is high. This results in high transient loads at the processor, leading to large queue buildups and high average response times. When N is large, the standard deviation of the probability distribution of the number of users in each cell is much smaller. This explains the lower average response times seen for $N = 100$.

The second observation regards the impact of the net flow rate of users, $N\lambda_r$, on average user agent response time. For example, in Figure 3b, with $N = 20$ and a load of 0.8, the average response time is 15.5 when the flow rate is 0.1 and the average response time is 7.03 when the flow rate is 10.0 (the average response time is 5.0 for the Never-migrate policy). Thus, as user movement rate decreases, we find that the average response time increases substantially.

At first glance, this effect that the flow rate has on the average response time seems inexplicable since the probability distribution of the number of users, P_i, is independent of the flow rate. It can be explained intuitively as follows. Note that the cell resident time $1/\lambda_r$ is inversely proportional to the flow rate. When the mobile users move more slowly, they spend longer periods of time in each cell. Even though two systems with different flow rates have the same probability distribution for the number of users, the system with a higher flow rate undergoes frequent changes in the number of users present in comparison to a system with a lower flow rate. Now, consider the case where the cell has $n > N$ users (and hence n user agents). Clearly, the load at the cell $n\lambda_e/\mu$ is higher than the average load. Now, when the mobile users move fast, there is a greater chance that the number of users (and hence the load) will decrease towards the average, N. This in turn, lets the processor "catch up" with the requests submitted during the overload situation. In order to maintain a constant probability distribution of the number of users, a system with a higher flow rate will make more frequent visits to saturated states (with $n > N$ users) but will also leave those states frequently, thereby limiting large queue buildups. On the other hand, when the mobile users slow their movement, the processors will spend longer periods of time in overloaded states before the load decreases due to departure of a mobile user. This will lead to larger queue buildups and, hence, larger average response times.

To summarize our observations we find that, with the average load constant, the average user agent response time is high when fewer users are present in each cell and when users move more slowly. In some cases, for example with a load of 0.8, $N = 20$ and $N\lambda_r = 0.1$, the average response time (15.5) is more than three times the average response time of a perfectly load balanced system (5.0). On the other hand, the average distance is directly proportional to the diameter of the network for the Never-migrate policy whereas the average distance is 0 for the Always-migrate policy. These differences in the values of the two primary performance metrics of these policies clearly argue for the need to examine more sophisticated migration policies which achieve a balance between the two metrics. This will be the subject of our study in the following section.

5 Analysis of Threshold Migration Policies

In this section, we discuss the two threshold-based migration policies in more detail and present their analysis. As before, we assume that the network is homogeneous and analyze a single cell using the decomposition approxima-

tion. The performance of these policies will be compared with the Always-migrate and the Never-migrate policies in the next section.

5.1 Count Policy

Recall that under the Count policy, user agents migrate to the destination cell along with the mobile user only if the number of user agents at the destination processor is less than a specified threshold, $T : N \leq T \leq L$.

Let P_r^{Count} denote the probability that a user returns to the cell containing its user agent given that the user performs a handoff. Then, the flow rate of user agents into a cell with m user agents is shown in [9] to be

$$I_{m,\cdot,\cdot}^{Count} = \begin{cases} N * \lambda_r - m * P_r^{Count} * \lambda_r, & 0 \leq m < T \\ 0, & m = T \end{cases}$$

Let P_T denote the probability that the cell has T agents. The flow rate of user agents out of a cell with m user agents is shown in [9] to be

$$O_{m,\cdot,\cdot}^{Count} = \begin{cases} m * \lambda_r * (1 - P_A), & 1 \leq m < T \\ T * \lambda_r * (1 - P_T), & m = T \\ 0 & \text{otherwise} \end{cases}$$

where $P_A = P_r^{Count} + (C - 1) * P_T/C$.

By analyzing the state transitions, it can be shown that [9] P_T satisfies a non-linear equation which can be solved using the method of bisection. Once P_T is computed, we can determine the different state transition rates and derive the infinitesimal generator Q for the system. We then follow the general analysis methodology presented in Section 4.2.1 to compute the average response time, R^{Count}.

In order to determine the average distance, let

$$P_d = Pr\{((v_i^{Count} = h) \cap (\Delta(g, h) = d))|(y_i^{Count} = g)\}$$

where v_i is the position of user after ith handoff, y_i^{π} is the position of agent after ith handoff under policy π, and $\Delta(\cdot)$ is the distance between two locations in the network.

Then

$$D^{Count} = \Sigma_{d=0}^{\infty} d * P_d$$

The average distance D^{Count} is shown in [9] to be

$$D^{Count} = \Sigma_{d=1}^{\infty} d * \Sigma_{i=1}^{\infty} \frac{(1-P_T)P_T^i \Sigma_{w(i) \in W(i)} Z(w(i),d)}{\gamma^i}. \quad (7)$$

where $Z(w(i), d)$ is 1 if the mobile user is at a distance d from the agent after the ith handoff of sequence $w(i)$ consisting of an unconstrained handoff and i constrained handoffs and $W(i)$ is the set of all handoff sequences with i handoffs, $|W(i)| = \gamma^i$

5.2 Distance Policy

Under the Distance policy, when the processor is not underloaded, only user agents whose users are at a distance greater than a threshold T are migrated.

Let P_r^{Dist} denote the probability that a user returns to the cell containing its user agent given that the user performs a handoff. Then, the flow rate of user agents into a cell with m user agents is shown in [9] to be

$$I_{m,\cdot,\cdot}^{Dist} = \begin{cases} \lambda_r(N - m * P_r^{Dist}), & 0 \leq m < N \\ \lambda_r(N - m * P_r^{Dist})X, & N \leq m < L \\ 0, & m = L \end{cases}$$

where X denotes the probability of migration when a processor is not underloaded.

The flow rate of user agents out of a cell with m user agents is shown in [9] to be

$$O_{m,\cdot,\cdot}^{Dist} = \begin{cases} m\lambda_r(1 - P_r^{Dist} - P_B), & 1 \leq m < N \\ m\lambda_r(1 - P_r^{Dist}X - P_B), & N \leq m \leq L \\ 0 & \text{otherwise.} \end{cases}$$

where P_{N^+} denotes the probability that the cell has N or more agents, $P(m \geq N)$ and $P_B = (1 - X)P_{N^+}$.

P_{N^+} can be computed by writing out the flow balance equations and solving the resulting non-linear equation. Once P_{N^+} is computed, we can determine the different state transition rates. We then follow the general approach presented in Section 4.2.1 to compute the average response time, R^{Dist}.

The equation for the average distance, D^{Dist}, has a similar form to equation (7) for D^{Count}.

$$D^{Dist} = \sum_{d=1}^{\infty} d \sum_{i=1}^{\infty} \frac{(1 - P_{N^+})P_{N^+}^i \sum_{w \in W(i)} Z^{Dist}(w, d)}{\gamma^i}. \tag{8}$$

The reader is referred to [[9],Appendix D] for its derivation.

6 Performance Comparison of Migration Policies

In this section, we use our analytic models to compare the performance of the Always-migrate, Never-migrate, Count, and Distance policies. First, we examine the fundamental trade-offs that exist between the Count and the Distance policies. We then compare the Count and the Distance policies with the Always-migrate and Never-migrate policies with the goal of minimizing a linear cost function.

Figure 4 plots the average distance, D^{π}, versus average response time, R^{π}, for the Count and Distance policies. We assume a 21x21 network with $\gamma = 8$ neighbors for each cell. As the parameters N, λ_r and λ_c/μ are varied, the shape of the graph remains similar [9].

The points in the graph are obtained by varying the threshold value for the Count and Distance policies. In the case of the Count policy, the threshold value increases from left to right in the graph while in the case of the Distance

policy, the threshold value decreases from left to right (lines shown to connect point values). Observe that the curves for the two policies intersect. Thus, there are regions in the graph where each policy performs better than the other. For example, the Count policy yields a lower mean response time than the Distance policy when the average distance is low. On the other hand, the Distance policy yields a slightly lower mean response time than the Count policy when the average distance is high.

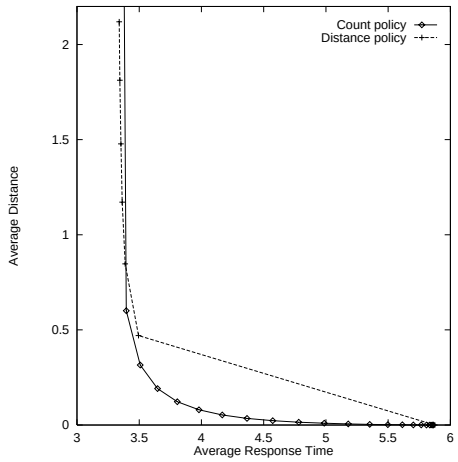


Figure 4: Comparison of Count and Distance migration policies with $N = 20, N\lambda_c/\mu = 0.7, N\lambda_r/\mu = 0.1$

From the spacing of point values in Figure 4, we conclude that the threshold in the Count policy provides a finer control when the average distance is kept low while the threshold in the Distance policy provides a finer control when the average response time is kept low.

In order to examine the impact of these observations in more detail, we now introduce a cost function, $F^\pi(\beta)$, as a linear combination of the average response time, R^π, and the average distance, D^π:

$$F^\pi(\beta) = R^\pi + \beta D^\pi$$

The goal of a migration policy, π, will be to minimize $F^\pi(\beta)$ for a given value of β. The parameter β represents the weight assessed to the distance relative to the average response time. A low value of β indicates that the response time metric is more critical to overall performance. This may be the case in a local area type application where the distance penalty associated with messages/data traveling between the user and the user agent is low. On the other hand, a high value of β represents the case where there is a high penalty associated with agents that are located far away from the mobile users they represent. This can represent the performance requirements of, for example, a wide-area application or a local-area application with large amounts of data flowing between the mobile user and the agent.

Figure 5 plots the function $F^\pi(\beta)$ against β in logscale for the two threshold policies and for the Always-migrate and Never-migrate policies.

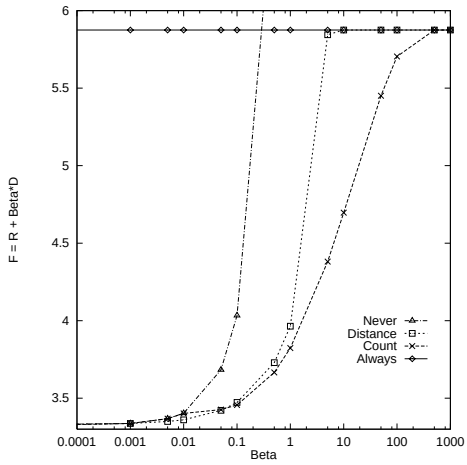


Figure 5: Comparison of the migration policies with $N = 20, N\lambda_c/\mu = 0.7, N\lambda_r/\mu = 0.1$

In the case of the Always-migrate policy, since the average distance, D^{Always} between the mobile users and their agents is zero, $F^{Always}(\beta) = R^{Always} = 5.875$ (equation 5). In the case of the Never-migrate policy, $D^{Never} = 6.98$ (equation 2) and $R^{Never} = 3.33$ (equation 1). Thus, $F^{Never}(\beta) = 3.33 + 6.98\beta$.

For the threshold policies, each datapoint is obtained analytically using the *optimal* threshold for that policy that minimizes the value of the cost function $F^{\pi}(\beta)$ for a given β. The function $F^{\pi}(\beta)$ is computed for a given policy π using the analysis presented in Section 4. The optimal threshold value that minimizes $F^{\pi}(\beta)$, is obtained through a binary search type algorithm over the threshold range for each of the polices. This procedure results in the optimum threshold because the two parameters of interest, R^{π} and D^{π}, both monotonically increase or decrease in their threshold values.

As the value of β is reduced, the optimal threshold value for each policy takes on a value that reduces user agent migrations. Thus, a value of $\beta = 0$ results in a choice of a threshold for the threshold policies such that there are no user agent migrations, mimicking the behavior of the Never-migrate policy. Similarly, a high value of β results in the choice of a threshold that tries to keep the user as close to the agent as possible. A very high value of β thus corresponds to the a case where user agent is always in the same cell as the mobile user, mimicking the behavior of the Always-migrate policies.

For moderate values of β, we observe that the threshold policies clearly outperform the Always-migrate and the Never-migrate policies. Further, the results show that there is a cross-over point between the Count and the Distance policy. For larger values of β, we find that the Count policy performs better than the Distance policy while for smaller values of β, the reverse is true. This behavior is exactly what we expect based on our earlier observations, when we found that the Count policy performs better when D^{π} is low (in other words, for large β) and the Distance policy performs slightly better when D^{π} is high (in other words, for small β).

In summary, we find that the Count policy is to be preferred when the optimal threshold value allows a significant amount of agent migrations (because of a high distance penalty) and the Distance policy is to be preferred when the

optimal threshold value restricts a large number of agent migrations (because of a very low distance penalty).

7 Analysis of Migration Policies: Heterogeneous Case

So far, we have restricted ourselves to examining migration policies in homogeneous networks. We now consider heterogeneous networks.

Heterogeneity can be in user agents (based on their processing requirements and job arrival rates) or in rate and direction of movement of mobile users. In order to incorporate heterogeneous user agents into our analysis, there are several simple strategies that can be adopted, such as a) maintain a threshold count for each type of agent, or b) weight each agent according to its processing needs and maintain a weighted threshold count. We believe that the results from the previous sections, namely, the qualitative differences between the count and distance policies will remain unchanged as we redefine thresholds to incorporate heterogeneous user agents. The above-mentioned strategies only serve to differentiate the performance between different types of user agents; this is orthogonal to differences between different types of migration policies.

We now extend our framework to encompass differences in the rate and direction of movement of the mobile users. We first describe our methodology for modeling heterogeneous traffic and then illustrate how the Always-migrate and Count policies can be analytically evaluated in this framework. The analysis of other policies can be performed in a similar manner.

While large portions of a wide-area wireless networks may be homogeneous, there will exist several smaller regions that display heterogeneous characteristics [12]. For example, isolated "hot spots" can occur in cities, stadiums etc. This heterogeneity can be captured only by modeling the cells in a heterogeneous region individually.

Consider a heterogeneous network with C cells and U users. Let the cells be labeled $0, 1, \cdots, C-1$. Heterogeneity in the network arises because of the differences in the flow rate of mobile users. To model heterogeneity, we use a $C \times C$ flow rate matrix, F, whose elements $f_{k,j}$ represent the flow rate of a mobile user moving from cell k to cell j. Clearly, the diagonal elements, $f_{k,k}$ are all zero. Also, to ensure that the markov models for each of the cells is ergodic, we require that $\sum_{j=0}^{C-1} f_{k,j}$ be the same for all the cells $k, 0 \leq k \leq C-1$. In the case of a homogeneous network, $f_{k,j}$ would be a constant value times $1/\gamma$ for all j cells that are neighbors of k and 0 otherwise. Under these modeling assumptions, we now evaluate the performance of the Always-migrate and the Count policies.

7.1 Always-migrate Policy

In the Always-migrate policy, the net flow rate of mobile users, $I^{Always}_{m,\cdot,k}$, into cell k, given that the cell has currently m user agents is

$$I^{Always}_{m,\cdot,k} = I^{Always}_{\cdot,\cdot,k} = \sum_{j=0}^{C-1} f_{j,k} M_j \tag{9}$$

where M_j denotes the average number of user agents (and also users) in cell j. The flow rate of users out of cell k, given that the cell has currently m user agents, is

$$O_{m,\cdot,k}^{Always} = \sum_{j=0}^{C-1} f_{k,j} m \tag{10}$$

Our first task is to compute M_k, the average number of user agents in cell k under the Always-migrate policy. This is easily computed by solving the following set of simultaneous linear equations:

$$\begin{array}{ccc} \sum_{k=0}^{C-1} M_k & = & U \\ \sum_{k=0}^{C-1} f_{k,0} M_k & = & \sum_{k=0}^{C-1} M_0 f_{0,k} \\ \sum_{k=0}^{C-1} f_{k,1} M_k & = & \sum_{k=0}^{C-1} M_1 f_{1,k} \\ \vdots & \vdots & \vdots \\ \sum_{k=0}^{C-1} f_{k,C-1} M_k & = & \sum_{k=0}^{C-1} M_{C-1} f_{C-1,k} \end{array} \tag{11}$$

Note that there are $C+1$ equations in equation set (11). The first equation is the conservation equation for the total number of users in the network. Out of the remaining C equations, one of the equations is redundant. Thus, we can solve for the C unknowns, M_k, from the remaining C equations.

Once we obtain the average number of users in each cell, the average response times for user requests in each cell can be easily computed using the input and output flow rates for each cell.

7.2 Count Policy

In the case of the Count policy, with a threshold of T_k in cell k, the flow rates into and out of cell k when m user agents are present is given by

$$I_{m,\cdot,k}^{Count} = I_{\cdot,\cdot,k}^{Count} = \sum_{j=0}^{C-1} f_{j,k} M_j \tag{12}$$

$$O_{m,\cdot,k}^{count} = \sum_{j=0}^{C-1} f_{k,j} m (1 - P_b(j)) \tag{13}$$

where $P_b(k)$ denotes the probability that a migration request would be blocked (denied).

The mean number of user agents in cell k, M_k, is given by

$$M_k = \sum_{i=0}^{T_k} i I_{\cdot,\cdot,k}^{Count} / O_{i,\cdot,k}^{Count}, k = 0, \ldots, C-1 \tag{14}$$

and the blocking probability in cell k, $P_b(k)$, is given by

$$P_b(k) = \frac{\frac{\{I_{\cdot,\cdot,k}^{Count}\}^{T_k}}{\prod_{i=0}^{T_k} O_{i,\cdot,k}^{Count}}}{\sum_{j=0}^{T_k} \frac{\{I_{\cdot,\cdot,k}^{Count}\}^{j}}{\prod_{i=0}^{j} O_{i,\cdot,k}^{Count}}}, k = 0, \ldots, C-1 \tag{15}$$

We now have two sets of C equations, with two sets of unknowns, $P_b(k)$ and M_k, where the $P_b(k)$'s are related to each other in a non-linear manner. We adopt a two-level iterative procedure to solve for the blocking probabilities (details in [9]). Once we compute $P_b(k)$ and M_k for every cell k, we can write out the infinitesimal generator for the user mobility behavior and compute the average response time for each cell. The average distance between the mobile user and the agent can be computed given $P_b(k)$ and the connection topology between the various cells.

7.3 Numerical Example

Let us consider an example network consisting of $C = 3$ cells and $U = 60$ users. We will illustrate the computation of the average response time and the average distance between mobile users and their agents for the Always-migrate and the Count policy.

Let the flow rate matrix be

$$F = \begin{vmatrix} 0.0 & 1.0 & 0.0 \\ 0.5 & 0.0 & 0.5 \\ 0.25 & 0.75 & 0.0 \end{vmatrix}$$

Note that a large fraction of the traffic from cells 0 and 2 flow into cell 1, thus creating a "hot spot" in cell 1. If we use the Always-migrate policy, the average number of users (and user agents) in each of the cells, obtained by solving equation set (11), are given by $M_0 = 17.647$, $M_1 = 28.235$ and $M_2 = 14.118$. Given that M_1 is twice M_2 (and hence the load at cell 1 is twice the load at cell 2), the Always-migrate policy is clearly not a good policy to adopt in such a system.

Using the Count threshold policy to balance the load in this system requires the determination of a set of three threshold values (for the three cells) which results in three average response time and average distance values. To examine the impact of thresholds in this example, we use a cost function, $F = \sum_{k=0}^{2}(R_k + D_k)/3$, where R_k denotes the average response time for user agent requests in cell k and D_k denotes the average distance values for users leaving cell k. We also take $\lambda_c = 0.035$ and $\mu = 1$ for each of the cells.

Figure 6 plots the cost function F versus the threshold in the hot spot cell (cell 1) for three different pairs of thresholds in cell 0 and cell 2. The value of F obtained using the Always-migrate policy is 30.89. Comparing this to the minimum value of $F = 3.77$ in Figure 6, we see that the Count policy results in a 88% improvement with respect to the Always-migrate policy. This corroborates our intuition that heterogeneity in networks can result in enormous imbalances in load.

Examining the effect of thresholds of the Count policy, the behavior of the cost function in Figure 6 with respect to the threshold T_1 can be explained as follows. Observe that as the threshold value T_1 is reduced from the right in the figure, the cost function also decreases. This is because cell 1 plays the role of a "hot spot" with more average users than the other cells. By limiting the number of agents in cell 1 where the load is the highest, while increasing the number of agents in the other cells where the load is lower, we reduce the average response times. Also, since the average distance does not go up appreciably (since we are dealing with a very small network), the value of the cost

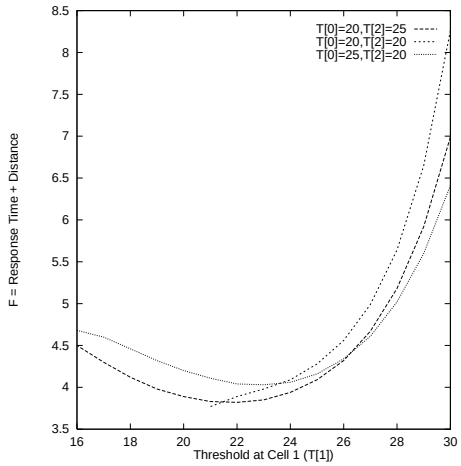


Figure 6: Count policy in heterogeneous networks

function starts to decrease. However, after a certain number of reductions in the value T_1, the burden of higher load shifts to one of the other cells which results in an increased average response time. By now, the average distance values have also gone up considerably since the amount of blocking in the overall system has increased. Thus, we see the value of the cost function increases for low values of T_1. Note that the threshold values must be chosen so that $T_0 + T_1 + T_2 > N$. This explains the reason why the plot corresponding to $T_0 = 20, T_2 = 20$ ends at $T_1 = 21$.

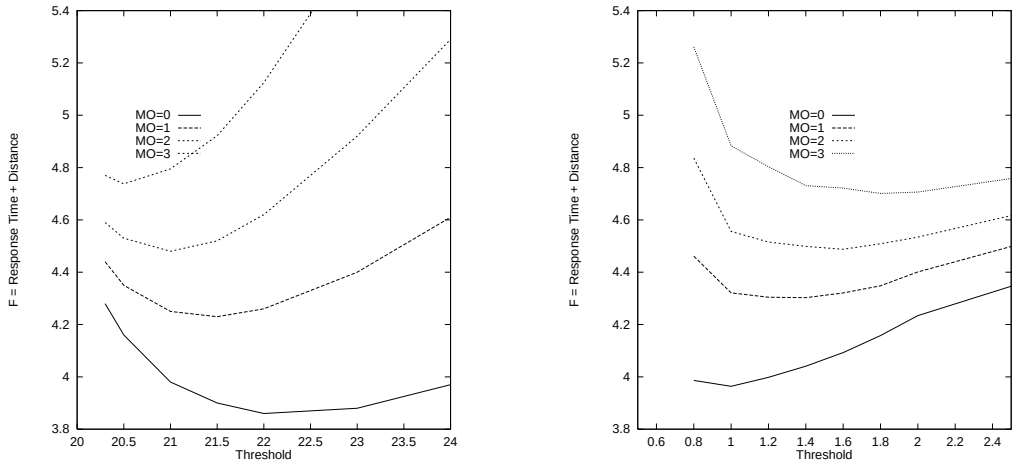
At present, we are not able to compute the optimal threshold values other than by a brute force approach. It would be interesting to examine the structural properties of the cost function and derive efficient algorithms for the computation of optimal thresholds in this case.

8 Migration Policies: Impact of Migration Overhead

In this section, we examine the impact of load balancing in the presence of processing costs due to the migration of the user agent. Capturing this migration cost in an analytical model leads to a four-dimensional markov process with the following modeled state variables in a cell: the number of user agents, the queued number of user agent jobs, the queued number of migration-into-the-cell jobs, and the queued number of migration-out-of-the-cell jobs. This makes the analysis intractable and thus, we examine the issue of migration overhead through simulation.

In the absence of migration overhead, we have observed that employing the Count and Distance policies result in significant reductions in the value of the cost function, $F(\beta)$, in comparison to the Always-migrate and Never-migrate policies. However, in the presence of migration overhead, the performance impact of these policies on $F(\beta)$ is not clear. This is because these policies differ in the way they restrict the number of user agent migrations. Note that the number of user agent migrations did not have an impact on the evaluation of these policies earlier. In the presence of migration overhead, user agent migration results in an increased load at the processor.

Unlike traditional process migration techniques, since we are migrating a specific process, we assume that the executable code is present at the target processor and accomplish the migration by transferring only a small amount



(a) Count policy

(b) Distance policy

Figure 7: F(1)=T+D in the presence of Migration Overhead

of dynamic state information of the user agent. The migration overhead is represented by $0.1 + 0.2 * MO$ to capture the fixed and variable cost of our process migration implementation [8] (variable cost represents the amount of state information migrated). We refer to [9] for details of the simulator.

Figure 7 plots the cost function $F^{\pi}(1) = T^{\pi} + D^{\pi}$ against the threshold for the Count and Distance policies and different values of migration overhead, MO (with $N = 20$, net user flow rate of 0.1, and net load of 0.7). The first important observation one can make from the figure is that the Count policy outperforms the Distance policy for low values of MO while the Distance policy outperforms the Count policy for $MO = 2$ and $MO = 3$. The second observation regards threshold values which minimizes the cost function F. Under the Count policy, the optimal threshold values moves to the left on the x-axis as MO increases. Under the Distance policy, the optimal threshold values moves to the right on the x-axis as MO increases. However, recall that a decrease (increase) in the threshold in the Count (Distance) policy results in fewer migrations. Thus, the optimal threshold values change under both these policies such that, as MO increases, the policy allows fewer migrations. Combining the above two observations, we find that the Count policy works better for low migration overhead systems (in which case the policy allows a lot of migrations) and the Distance policy works better for high migration overhead systems (in which case the policy significantly restricts migrations). This reinforces our earlier conclusion that the Distance policy is to be preferred when the optimum value of the threshold severely restricts agent migrations and the Count policy is to be preferred otherwise.

9 Conclusion

In this paper, we considered a fundamental question in the design of wireless networks that employ network-based user agents as proxies for mobile users: what are the parameters and trade-offs that dictate the design of migration policies for the user agents?

We began by developing a general framework for studying and analyzing the performance of migration policies. The policies were compared using two performance metrics: the average response time for a user request and the average distance separating mobile users from their agents. We first showed that the two basic policies, Always-migrate and Never-migrate, resulted in poor performance thereby motivating the need for more sophisticated policies. We therefore looked at two threshold based policies; a Count policy that limits the number of agents in each cell, and a Distance policy that allows user agent migrations to processors that are not underloaded, only when the user is separated by more than a specific distance from the user agent. We studied these policies in the context of both homogeneous networks and heterogeneous networks, and in the presence and absence of processing overhead due to migration. Over a wide range of system parameters, the Count and the Distance policies were shown to outperform the Always-migrate and the Never-migrate policies. We further showed that the Count policy is to be preferred when it is necessary to allow a significant amount of agent migrations (e.g., because of a high distance penalty or a low migration overhead) and the Distance policy is to be preferred when it is necessary to severely restrict agent migrations (e.g., because of very low distance penalty or a high migration overhead).

References

[1] N. Adams, R. Gold, B. Schilit, M.Tso and R.Want, "An Infrared Network for Mobile Computers," in Proceedings of the Usenix Mobile and Location-Independent Computing Symposium, August 1993.

[2] E. Amir, S. McCanne, R. Katz, "The Media Gateway Architecture: A Prototype for Active Services," in Proceedings of ACM SIGCOMM'98.

[3] D. L. Eager, E.D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," in IEEE Transactions on Software Engineering," Vol 12, No. 5, 1986.

[4] My T. Le, F. Brughardt, S. Seshan and Jan Rabaey, "InfoNet: The networking infrastructure of Infopad," Proceedings of Compcon, Calif., Mar. 1995

[5] M.Litzkow and M.Solomon, "Supporting checkpointing and process migration outside the Unix Kernel," in Usenix Winter Conference, San Francisco, California, 1992.

[6] M.F. Neuts, "Matrix Geometric Solution in Stochastic Models: An Algorithmic Approach," Dover Publications, New York, 1994.

[7] C.E. Perkins, "IP Mobility Support," Internet Request for Comments RFC 2002, October 1996.

[8] R. Ramjee, T. La Porta, M. Veeraraghavan, "The use of Network-based Migrating User Agents for Personal Communication Services," in IEEE Personal Communications Magazine, Vol. 2, No. 6, Dec 1995.

[9] R. Ramjee, "Supporting Connection Mobility in Wireless Networks," Ph.D. Dissertation, University of Massachusetts, 1997.

[10] Y.-B. Lin and V.W. Mak, "Eliminating the Boundary Effect of a Large-Scale Personal Communication Service Network Simulation," in ACM Transactions on Modeling and Computer Simulation, Vol. 4, No. 2, April 1994, pp 165-190.

[11] 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,* Vol 11. No 6. (August 1993), pp 911-917.

[12] T.-S. Yum and W.-S. Wong, "Hot-spot traffic relief in cellular systems," *IEEE Journal on Selected Areas in Communications,* 11(6):934–940, Aug. 1993.