Minimizing queuing delays and number of messages in mobile phone location

David Goodman [a], P. Krishnan [b] and Binay Sugla [b]

[a] WINLAB, Rutgers University, Box 909, Piscataway, NJ 08855, USA
[b] Bell Laboratories, 101 Crawfords Corner Road, Holmdel, NJ 07733, USA

Abstract. The mobility of phones in a cellular or Personal Communication Services (PCS) environment introduces the problem of efficiently locating the called phone. In this paper, we present an analysis of the delay and number of messages transmitted in different sequential and parallel search strategies, considering for the first time the issue of queuing on radio paging channels. Our analysis shows that parallel search may not reduce the time to find a mobile phone if the parameters of the system are unfavorable. We also develop an efficient algorithm for searching with minimum expected number of messages when the location of the phone is given by a probability vector.

1. Introduction

In a traditional phone network, each phone is associated with a known geographical location. The numbering scheme for these fixed phones takes advantage of their known location. A relatively simple mapping exists from the telephone number to its geographical location. The introduction of the 800 number service utilized an indirection in this mapping while still exploiting the property that the final number is placed at a known geographical location. The 700 number service, first introduced by AT&T, exploits the fact that the called number may be mapped into one of several fixed numbers using a logic and order specified by the customer. The 500 number service is more advanced in that the customer can have his/her calls forwarded to (virtually) any phone; however the mapping between the customer's 500 number and a standard number must be provided by the customer, e.g., by using a touch-tone phone. The techniques used in the services mentioned above are inadequate for the mobile environment since the location of the final destination (viz. the called phone) is unknown and must be determined before the call is completed.

In response to this need, several schemes have been employed and suggested. A centralized paging scheme in which the called phone number is broadcast and the called phone responds is inefficient in the use of radio bandwidth. In the more recent search techniques, the basic unit of paging is the cell level. The problem now is to design an efficient search algorithm such that the relevant costs are minimized.

A number of research papers address facets of the mobile location problem. The problem is complex because there are several performance criteria including costs of reporting, recording, and retrieving the locations of mobile phones, costs of searching for mobile phones when calls arrive, the probability of a successful search, and delays in finding phones. Tracking and search costs include radio channel occupancy, transmissions in fixed networks, and database transactions. The overall complexity is also important because a scheme that requires a highly distributed real-time system may introduce problems of its own, particularly with respect to reliability. Previous papers [1–5,7,8] address network architecture issues, database structures, and tradeoffs between registration and paging costs.

This paper focuses on the search process. We assume that the system has accurate knowledge that a phone is in a "location area" which consists of a collection of cells. We then examine sequential search strategies for determining the cell in which the phone is located. The search procedure consists of sending paging messages to a group of cells in the location area and waiting for a response from the phone. If no response arrives within a fixed waiting period, the network sends paging messages in another group of cells, and again waits for a response. The procedure continues until the phone responds to a paging message. The quality criteria that we examine are the search delay and the total number of messages transmitted.

An important issue is the queuing delay of paging messages on radio channels. Our probabilistic analysis reveals that parallel search may not reduce the time to find a mobile phone if the parameters of the system are unfavorable. We also develop an efficient algorithm for paging to minimize the total number of messages when we know the probability of finding a phone in any specific cell.

Rose and Yates inform us that they have also independently come up with a similar queuing delay analysis [9] and report an independent analysis of paging to minimize total number of messages [10]. Our goal in this

paper is to study the design implications of our analysis taking into account both the delay incurred and number of messages transmitted. The delay analysis focuses on uniformly distributed traffic and includes the effect of a timeout at each cell. In [9], the authors analyze delay as a function of the mean of the ordered distribution that describes the uncertainty about the location of the mobile phone. For the problem of developing an efficient algorithm for paging to minimize the number of messages when we know the probability of finding a phone in any paging cell, our emphasis in this paper is on the computational complexity of the solution, while in [10] the authors deal with methods for different types of distributions.

In [6], Madhavapeddy et al. propose methods to empirically compute the probability of a mobile phone being in any paging cell using the registration history. They also develop algorithms to minimize the expected number of messages while searching for a mobile phone; we compare and contrast our algorithms with the ones in [6] in section 4.

The rest of the paper is organized as follows. In section 2 we present our model. In section 3, we introduce and present an analysis of sequential and parallel schemes used for locating a mobile phone. The simple analysis in section 3.1 raises important questions about the issue of queuing delays. In section 3.2, we analyze the effect of queuing on radio paging channels, assuming a Poisson call-arrival. In section 4, we develop an efficient paging strategy to minimize the total number of messages sent in locating the mobile phone when the possible locations of the phone are specified by a probability vector. We conclude in section 5 with directions for future research.

2. The model, definitions, and notation

We denote the number of base stations by N_B. The called telephone can be in any one of the N_B cells served by the base stations. In the basic search technique, the search is divided into *stages*, where in the jth stage, k_j stations that have not searched for the phone in earlier stages query for the phone; the search stops when the phone is found, or all stations have searched for the phone. The stations that page in a stage form a *paging group*. The maximum number of paging groups is denoted by N_G. (In other words, the paging groups partition the set of N_B stations into N_G subsets.) The case when $N_G = N_B$ is called *polling*, and the case when $N_G = 1$ is called *flooding*.

Let p_i, for $1 \leqslant i \leqslant N_B$, be the probability that the phone is in the service area of base station i, let S be the random variable denoting the number of stages in the search for the phone, and let M be the random variable denoting the total number of query messages sent in locating the phone. We define the *uniform case* as a situation in which

a mobile phone is equally likely to be in any cell in the location area. Furthermore, we divide the location area into N_G equal-sized paging groups, each containing k cells. In terms of our notation, we have for the uniform case,

$$p_i = \frac{1}{N_B} \qquad \text{for all } 1 \leqslant i \leqslant N_B,$$

and

$$k_j = k = \frac{N_B}{N_G} \qquad \text{for all } 1 \leqslant j \leqslant N_G.$$

Two important quantities we study in this paper are the *delay* and the *number of messages transmitted* in searching for a mobile phone. In this paper we assume that the number of messages transmitted is equal to the number of base stations that page before the phone is located. In general, the delay (or time taken in locating the phone) is a function of the number of paging groups, the delay in the wireless channel, the timeout variable (i.e., the amount of time a station waits before deciding that the phone is not present in its paging area), the delay in the wired network, and the computational time at each base station and the switching network. Typically, the delay in the wired network and the computational delays are comparatively negligible.

3. Analysis of search techniques

In this section, we study the uniform case in detail. This corresponds to the situation when the customer with the phone moves randomly and hence $p_i = 1/N_B$, for all i. We discuss in section 4 the situation when the p_i's are all not necessarily $1/N_B$ and the probability vector p is known beforehand. Some more discussion of the non-uniform case appears in Appendix A.

3.1. Basic sequential and parallel search techniques

The primary quantities of interest in analyzing a search strategy are the expected and maximum number of stages (denoted by $E(S)$ and $\max(S)$ respectively), and the expected and maximum number of query messages (denoted by $E(M)$ and $\max(M)$, respectively).

It is obvious that $\max(S) = N_G = N_B/k$, and $\max(M) = N_B$. Clearly $E(S) = \sum_{j=1}^{N_B/k} j \cdot (1/N_G)$, since with probability $1/N_G$, we will have j stages. Hence, $E(S) = (N_B + k)/2k = (N_G + 1)/2$. Since in each stage k messages are sent, $E(M) = k \cdot E(S) = (N_B + k)/2$.

Table 1 summarizes this information, with the values of $E(S)$, $\max(S)$, $E(M)$, and $\max(M)$, for general fixed k, and for polling and flooding. The tradeoffs are easy to notice: if the delay in finding a phone is proportional to the number of stages, flooding finds the phone quickly with more messages, while polling takes longer but sends fewer messages.

Method	$E(S)$	$\max(S)$	$E(M)$	$\max(M)$
General k	$\dfrac{N_G + 1}{2}$	N_G	$\dfrac{N_B + k}{2}$	N_B
Polling $(k = 1)$	$\dfrac{N_B + 1}{2}$	N_B	$\dfrac{N_B + 1}{2}$	N_B
Flooding $(k = N_B)$	1	1	N_B	N_B

3.2. Analysis of search techniques with queuing delays

The tradeoff observed from the analysis in section 3.1 assumes that the delay in finding a phone in any stage is $O(1)$. However, in practice, paging messages will get queued at a station before being broadcast, because of the volume of calls. This implies that the response time delay associated with finding a mobile phone is not proportional to the number of stages anymore. We now analyze search techniques taking into account the effect of queuing delays at the stations. Our analysis sheds light on important design issues related to the number of stations that must page at each stage.

Our goal in this section is to determine an expression for the delay for the uniform case; this relationship will shed light on how to organize the paging. In Theorem 1, we determine an expression for the expected queue length at a station that is relevant for delay analysis. In Theorem 2, we derive an expression for the expected delay in paging based on our expression for the expected queue length. We then use this expression for the expected delay to derive in section 3.3 an interesting result that affects the design of paging systems.

To obtain our expression for the queue length, we make some simplifying assumptions about the system, which are reasonable in our current setting.

Assumption 1. We assume the uniform case as described in section 2. We assume that the entire delay bottleneck in any stage is in the radio channels. Each channel is an $M/M/1$ queue with average capacity of μ messages per unit time (i.e., each channel services broadcast requests at an average rate of μ messages per unit time). We assume that call arrivals are Poisson with aggregate arrival rate of ϕ calls per unit time. We assume that the k stations in each paging group are chosen at random from stations that have not yet paged. In other words, the paging groups are not fixed [1]. Assuming that the call arrivals are uniform over the N_B base stations, the arrival rate on each paging channel is $\phi S / N_G$. The

[1] We make this assumption for mathematical simplicity. In practice, the paging groups may be fixed before-hand, which introduces a mathematical dependency between the queue lengths at different stations.

average arrival rate on each paging channel is $\lambda(N_G) = \phi \cdot E(S)/N_G$. From Table 1, $\lambda(N_G) = \phi \cdot (N_G + 1)/2N_G$. (Notice that in flooding $\lambda(N_G) = \lambda(1) = \phi$, while in polling $\lambda(N_G) = \lambda(N_B) = \phi \cdot (N_B + 1)/2N_B$.)

We denote by $\rho(N_G)$ the quantity $\lambda(N_G)/\mu$. For notational convenience, we denote $\lambda(N_G)$ by λ and $\rho(N_G)$ by ρ; it should be borne in mind that for any fixed ϕ, the quantities λ and ρ are functions of N_G.

Let Q_i be the random variable denoting the queue length at base station (paging station) i. The delay induced by a queue that has ℓ outstanding paging requests is ℓ/μ. Assume without loss of generality that in the jth stage we request stations $1, 2, \ldots, k$ to broadcast paging messages. Intuitively, the delay D_j in the jth stage is closely related to the *maximum queue length* amongst the k base stations that page in that stage; i.e., D_j, for $1 \leqslant j < S$ is related to the random variable $G_j = \max(Q_1, Q_2, \ldots, Q_k)$. To get an expression for the expected delay, we would like to derive an expression for $E(G_j)$. We proceed to determine $E(G_j)$ from an expression for $\Pr(Q_i \leqslant \ell)$ as follows.

By $M/M/1$ queuing theory, the queue length Q_i at base station i is a random variable with geometric distribution having parameter $\rho = \lambda/\mu$; i.e., $\forall i$, $\Pr(Q_i = \ell) = (1 - \rho)\rho^\ell$. Hence,

$$\Pr(Q_i \leqslant \ell) = 1 - \rho^{\ell+1}.$$

By Assumption 1, the Q_i are i.i.d. Therefore,

$$\Pr(G_j \leqslant \ell) = (\Pr(Q_i \leqslant \ell))^k = (1 - \rho^{\ell+1})^k. \quad (1)$$

Further,

$$E(Q_i) = \frac{\rho}{1 - \rho}. \quad (2)$$

Since $E(G_j) = \sum_{\ell \geqslant 0} \ell \cdot \Pr(G_j = \ell) = \sum_{\ell \geqslant 0} \Pr(G_j > \ell)$, we have from (1) that

$$E(G_j) = \sum_{\ell \geqslant 0} (1 - (1 - \rho^{\ell+1})^k). \quad (3)$$

Expanding $(1 - \rho^{\ell+1})^k$ in (3) and interchanging the order of summation, we get

$$E(G_j) = \sum_{m=1}^{k} \binom{k}{m} (-1)^{m-1} \sum_{\ell > 0} \rho^{m\ell},$$

which yields the following theorem.

Theorem 1. Under the conditions listed in Assumption 1, $E(G_j)$, the average maximum queue length amongst the k stations that page in the jth stage is given by

$$E(G_j) = \sum_{m=1}^{k} \binom{k}{m} (-1)^{m-1} \frac{\rho^m}{1 - \rho^m}. \quad (4)$$

We are now ready to derive an expression for $E(D)$,

the expected delay in locating a mobile phone. We denote the expected delay of the jth stage by $E(D_j)$. Since the k stations to page in the jth stage are chosen at random, it implies that $E(G_j)$ is the same for all j, where $1 \leqslant j < S$; with a slight abuse of notation, we will denote this average by $E(G)$. Similarly, by Assumption 1, since the Q_i's are i.i.d., we will use the notation $E(Q)$ to denote the expected queue length at any base station.

When a call arrives at a switch, the system sends messages to the paging group. The request eventually finds its way to the front of the queue and is broadcast. The station either gets a positive acknowledgement (say, immediately), or waits a fixed amount of time W before it times out and assumes the mobile phone is not found. We now compute the delay for the search strategy.

Let us assume the mobile phone is eventually found. (The case when the phone cannot be located is less interesting; the analysis is similar, and is omitted.) If

the mobile phone is found in stage s, the expected delay in this stage is $E(D_s) = E(Q)/\mu$, where $E(Q) = E(Q_i)$ is as in (2). (If station i locates the mobile phone, the expected delay is $E(Q_i)/\mu = E(Q)/\mu$.) For an unsuccessful stage j, for $1 \leqslant j < s$, the expected delay is $E(D_j) = E(G)/\mu + W$, since it takes $E(G)/\mu$ time to exhaust the queue, and the last station that broadcasts the requests waits W units of time before timing out. Hence the average delay is

$$E(D) = \frac{E(Q)}{\mu} + (E(S) - 1) \cdot \left(\frac{E(G)}{\mu} + W \right),$$

where S is the number of stages. Substituting for $E(S)$ from Table 1, and for $E(Q)$ from (2) we get the following theorem.

Theorem 2. For any given load $\rho = \rho(N_G) = \lambda(N_G)/\mu$, under the conditions of Assumption 1, the expected

(a) $\rho(1) = \phi/\mu = 0.995$

(b) $\rho(1) = \phi/\mu = 0.95$

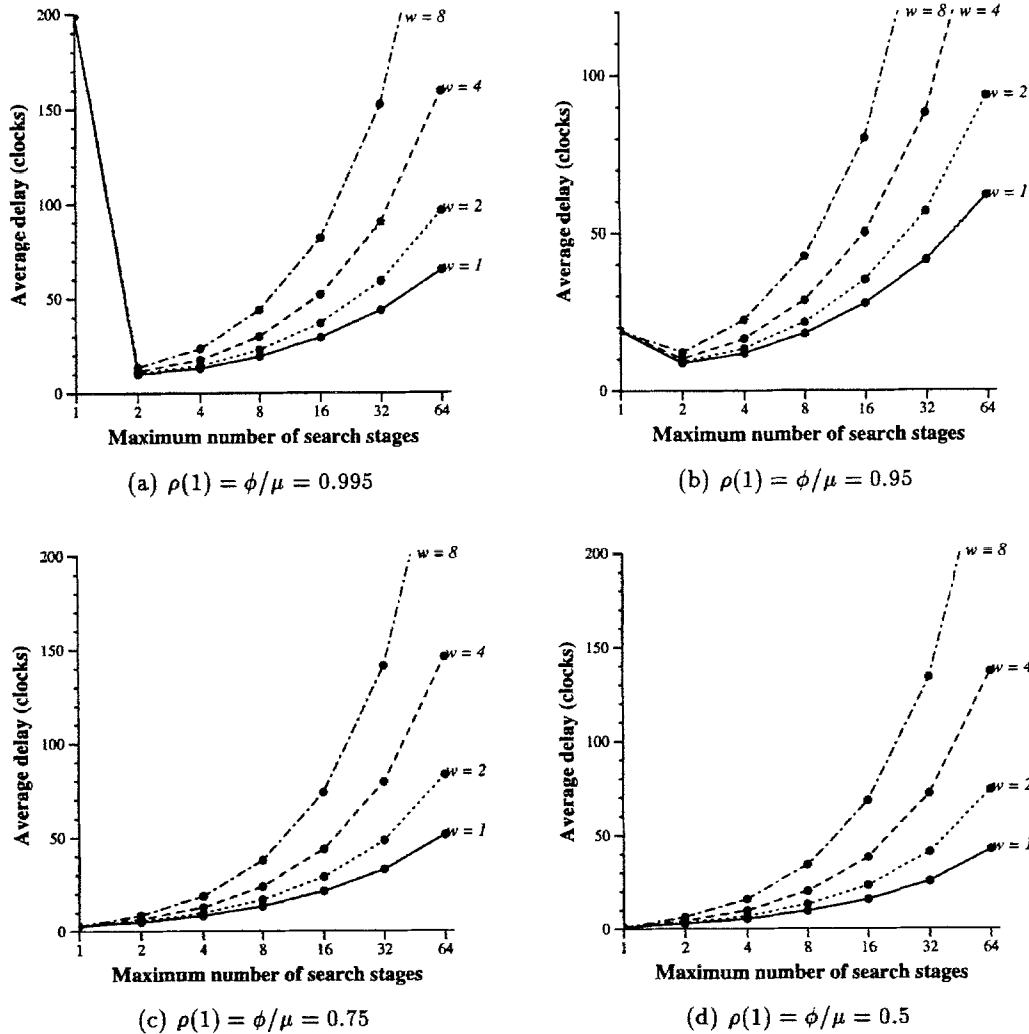(c) $\rho(1) = \phi/\mu = 0.75$

(d) $\rho(1) = \phi/\mu = 0.5$

Fig. 1. Average delay vs. the maximum number of paging groups N_G for four different values of load $\rho(1) = \phi/\mu$. The number of base stations is $N_B = 64$. The four different curves in each graph correspond to different values of the waiting time w. The x axis is logarithmic to better illustrate the interesting portions of the graph corresponding to small N_G. Portions of the graph corresponding to a high delay have been cropped. As seen from (a) and (b), the delay reduces dramatically in going from one to two paging groups in the presence of substantial load.

delay $E(D)$ in locating a mobile phone in the presence of queuing delays is given by

$$E(D) = \frac{\rho}{\mu(1-\rho)} + \left(\frac{N_G - 1}{2}\right) \cdot \left(\frac{E(G)}{\mu} + W\right),$$

when the phone is eventually found, where N_G is the number of paging groups, $E(G) = E(G_j)$ is as in Theorem 1, and W is the maximum waiting time for a response from the phone.

Theorem 2 gives us an analytical handle for the paging delay in the presence of queuing delays. To better understand the design implications of this analysis, we now study how the average delay varies with the number of search stages and the system loading factor $\rho(1) = \phi/\mu$.

3.3. Design implications of queuing delay analysis

In this section we look at numerical results based on the analysis in section 3.2. The channel loading variable $\rho(N_G)$ is a dimensionless quantity, and is the ratio of the total call arrival rate $\lambda(N_G)$ to the service rate μ on each paging channel. The aggregate arrival rate of calls is ϕ messages per unit time. Recall from Assumption 1 that $\lambda(N_G) = \phi \cdot (N_G + 1)/2N_G$.

We study the quantity $E(D) \times \mu$ (rather than just $E(D)$). The rationale behind this is simple: the quantity μ is fixed for the system by the deployed hardware, and multiplying by μ only scales the units. Our delay is therefore measured in the number of messages; we call our unit of delay a *clock*, where one clock is the time required to send one message on the paging channel. Similarly the *waiting timeout* $w = W\mu$ is the number of clocks spent waiting for a response.

In Fig. 1, for a fixed load $\phi/\mu < 1$, we plot the average delay as a function of the number of paging groups N_G. (In all four subfigures of Fig. 1, the number of base stations $N_B = 64$. The subfigures differ in the amount of congestion on the paging channel.) The graphs show that when the paging channel is not heavily loaded ($\phi/\mu < 0.95$, for example), flooding (corresponding to $N_G = 1$) results in the shortest delay. However, when paging channels face congestion, flooding places a high volume of messages on the paging channels. This results in very high queuing delays. In going from flooding to a two-stage search, we add to the average waiting time. However, this is more than compensated by the reduction in queuing time. Adding further stages beyond two is generally counter-productive. The increased waiting time is more significant than further reductions in queuing. When considered along with the total number of messages sent (see Table 2), perhaps $N_G = 4$ is a good value for the number of paging groups – this reduces the total number of messages (relative to $N_G = 1$) from N_B to $0.625N_B$ in addition to providing small delay.

Fig. 2 graphs the average delay as a function of chan-

Table 2
Average number of messages $E(M)$ vs. the number of paging groups N_G, based on equations from Table 1.

N_G	$k = N_B/N_G$	$E(M) = (N_B + k)/2$
1	N_B	N_B
2	$N_B/2$	$0.75N_B$
3	$N_B/3$	$0.66N_B$
4	$N_B/4$	$0.625N_B$
5	$N_B/5$	$0.6N_B$
$\vdots$	$\vdots$	$\vdots$
N_B	1	$\approx 0.5N_B$

nel load ϕ/μ. The graph illustrates dramatically that while the delay blows up for single stage paging, it remains manageable for two or more stages. Although not obvious, this is not surprising, since in a one stage search, the paging channels get inundated with messages which blows up the delay, while having two search stages reduces the messages more than linearly, as given by the expression for $E(G)$ in Theorem 1.

An alternative way of looking at the problem is to see what is the maximum single stage loading (ϕ/μ) that can be supported for a specific value of expected delay for different values of N_G. Fig. 3 plots for different values of d, the minimum ϕ/μ such that the average delay exceeds d vs. N_G. As we see from the graph, for a fixed d, the sustainable load increases with N_G, but beyond a point, the delay introduced by multiple stages predominates.

4. Minimizing $E(M)$ when probabilities are known

In this section we consider the general (non-uniform) case when the phone is not necessarily equally likely to be found in any of the N_B paging cells. Instead, we are given a probability vector $p = (p_1, p_2, \ldots, p_{N_B})$ such that p_i is the probability that the phone is in the paging area
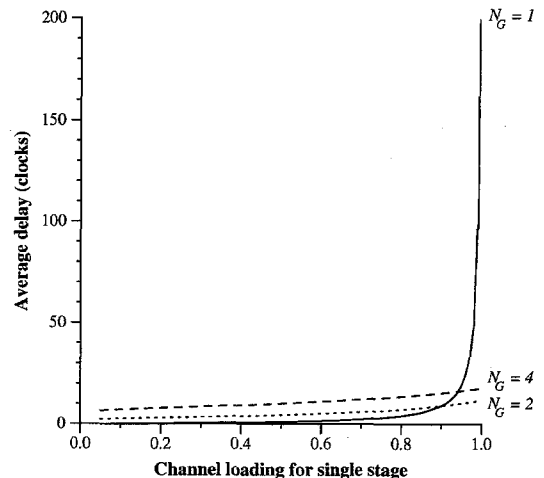

Fig. 2. Average delay vs. ϕ/μ. The curve for $N_G = 1$ has been cropped at a delay of 200. The number of base stations is $N_B = 64$.
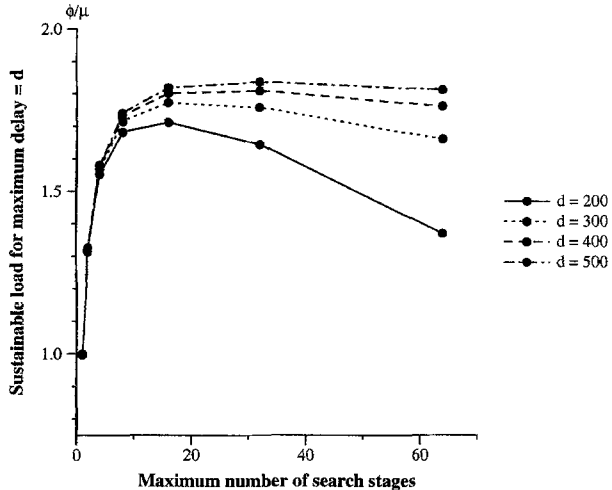
Fig. 3. The minimum single channel loading ϕ/μ such that delay exceeds d plotted against N_G, for four different values of d. The waiting timeout $w = 4$, and the number of base stations $N_B = 64$.

of base station i. (These probabilities could be estimated in practice by using the technique from [6], for example.) In this scenario, our goal is to find an efficient algorithm that groups the N_B stations into at most N_G paging groups to minimize $E(M)$, the average number of broadcast messages [2].

For this problem, we study two cases: We first study the special case when $N_G = 2$ in detail and develop an algorithm *Divide2* that partitions the N_G stations optimally and efficiently into two paging groups. Algorithm *Divide2* is based on "Algorithm 2" from [6], with a couple of simple enhancements. The $N_G = 2$ case is particularly interesting since our delay analysis from section 3.3 suggests that we gain a lot in going from $N_G = 1$ to $N_G = 2$ with respect to queuing delay at high loads. It is also of great interest to practitioners given its simplicity. We then develop a polynomial time algorithm *Group* that minimizes the expected number of messages for the general case when N_G is arbitrary.

Without loss of generality, assume $p_1 \geqslant p_2 \geqslant p_3 \geqslant \cdots \geqslant p_{N_B}$. Since the p_i's are not necessarily equal, each paging group need not have $k = N_B/N_G$ elements. Let paging set B_j be the set of k_j base stations that page in the jth stage. We denote $\sum_{i \in B_j} p_i$ by P_j. By the definition of expectation, it follows that

$$E(M) = \sum_{j=1}^{N_G} P_j \cdot \left(\sum_{m=1}^{j} k_m \right). \tag{5}$$

We make some interesting observations that form the basis for our algorithms *Divide2* and *Group*. We first show that there is a configuration that minimizes $E(M)$

such that for any $1 \leqslant i \leqslant N_B - 1$, stations i and $i + 1$ are in the same paging set B_j or in consecutive paging sets (i.e., station i is in set B_j and station $i + 1$ is in set B_{j+1}). In other words, we will never have a station ℓ page later than station i if $p_\ell > p_i$. (This fact is also noted in [6,10].)

Lemma 3. Let the sets B_1, B_2, $\ldots B_{N_G}$ be a partition of the N_B stations into N_G sets such that $E(M)$ is minimized. If station $i \in B_j$, then no station ℓ such that $p_\ell > p_i$ can be in set B_n, where $n > j$.

Proof. The proof is by contradiction. If a station ℓ as described in the lemma exists, we show that moving ℓ to B_j and moving i to B_n strictly reduces $E(M)$: by writing out the expressions for $E(M)$ in the two cases, we see that $E(M)$ changes by $(p_i - p_\ell) \cdot (B_{j+1} + B_{j+2} + \cdots + B_n)$, which is < 0, since $p_i < p_\ell$. $\square$

We now show that in the configuration that minimizes $E(M)$, there are indeed N_G paging sets (and not $N'_G < N_G$ paging sets).

Lemma 4. Let the sets B_1, B_2, $\ldots B_{N_G}$ be a partition of the N_B stations into $N_G \leq N_B$ sets such that $E(M)$ is minimized. Then $k_j > 0$, for $1 \leq j \leq N_G$.

Proof. The proof is by contradiction. Let B_1, B_2, $\ldots$, B_n be a partition that minimizes $E(M)$, and let $n < N_G$, where $|B_j| > 0$, $1 \leq j \leq n$. Let $B_m = \{i_1, i_2, \ldots, i_r\}$, where $k_m > 1$. (Trivially, such a set exists.) It is easy to verify that leaving B_j, $1 \leq j < m$ unchanged, renaming B_j to be B_{j+1} for all j from n down to $m + 1$, setting $B_m = \{i_1\}$, and $B_{m+1} = \{i_2, \ldots, i_r\}$, gives us $m + 1$ sets and $E(M)$ is reduced. $\square$

Notice from Lemmas 3 and 4 that by setting $N_G = N_B$, we essentially want to sort the probability vector; hence the lowerbound of $\Omega(N_B \log N_B)$ operations on sorting N_B numbers (in the comparison model) holds for our problem too.

4.1. Algorithm Divide2: The case when $N_G = 2$

When the number of paging groups is 2, we want to divide the N_G stations into two sets B_1 and B_2 so that $E(M)$ is minimized. Based on Lemmas 3 and 4, we see that we can first sort the N_B probabilities, and "test" the $N_B - 1$ possible positions to determine which position gives us two sets that minimize $E(M)$. We can do this by iteratively calculating the value of $E(M)$ for each k, for $1 \leqslant k \leqslant N_B - 1$, when the first k stations are in set B_1, and choosing the k that minimizes $E(M)$.

"Algorithm 2" from [6] computes B_1 and B_2 by observing the following property:

If B_1 has the first $k - 1$ stations, and $P = \sum_{i=1}^{k-1} p_i$,

by adding the kth station to set B_1, we have $\Delta E(M)$, the change in $E(M)$, as

$$\Delta E(M) = P - (N_B - k) \cdot p_k . \tag{6}$$

We need to increase k only as long as $\Delta E(M) < 0$.

Since $E(M)$ when taken as a function of k, the number of elements in set B_1, is convex, the above technique provides us with an optimal partitioning of the stations into two paging groups.

We make two simple observations below that can be used in conjunction with "Algorithm 2" from [6].

1. In our iterative process of computing the sets B_1 and B_2, let $|B_1| = k - 1$, and let $P = \sum_{i=1}^{k-1} p_i$ be the sum of the probabilities of the $k - 1$ elements in B_1. If $P + p_k \leqslant 1/2$, the kth station can be added to set B_1. This is because $p_k \leqslant p_{k-1} \leqslant P/(k-1)$, since p is sorted. By virtue of p being sorted, and $P + p_k \leqslant 1/2$, we have $p_k \cdot (N_B - k) \geqslant \sum_{i=k+1}^{N_B} p_i \geqslant 1/2$. Hence, $\Delta E(M) \leqslant 0$.

2. The convexity of $E(M)$ as a function of k combined with our knowledge that p is sorted implies that in the optimal breakup, $k \leqslant (N_B + 1)/2$. This is because we know that $p_k \leqslant P_{k-1}/(k-1)$; from (6) it follows that $\Delta E(M) \geqslant 0$ for $k \geqslant (N_B + 1)/2$.

The above observations provide us with algorithm *Divide2* described in Table 3. Notice that Observation 2 does not improve the time complexity of the strategy, but helps us in getting a better bound for the maximum amount of time it will take to do the partitioning; in particular, the algorithm takes at most $(N_B + 1)/2$ iterations after sorting the vector p.

Table 3
Algorithm *Divide2* based on "Algorithm 2" from [6] to partition N_B stations into two paging groups to minimize $E(M)$.

Algorithm *Divide2*

```
begin
    P := p₁;
    j := 2;
    while (P + pⱼ < 1/2) do
    begin
        P := P + pⱼ;
        j := j + 1;
    end
    for i = j, ..., (N_B + 1)/2 do
    begin
        ΔE(M) = P - (N_B - i) · pᵢ;
        if (ΔE(M) > 0) then
            B₁ := {1, 2, ..., i - 1};
            B₂ := {i, i + 1, ..., N_B};
            exit;
        else
            P = P + pᵢ;
        endif
    end
end
```

An interesting question in this case is: How unbalanced in terms of probability can sets B_1 and B_2 be in the optimal partitioning? We discuss this issue briefly in Appendix A.1.

4.2. Algorithm Group

In this section we look at the case when N_G is arbitrary. Intuitively, Lemmas 3 and 4 say that to get an optimal partitioning, we can sort the probabilities in p and then put $N_G - 1$ "marks" into this sorted list. There exists (at least) one set of marks which minimizes $E(M)$. The question now is one of determining this optimal set of marks efficiently.

An obvious generalization of algorithm *Divide2* to the case when $N_G > 2$ is a greedy method that moves elements between adjacent sets as long $E(M)$ decreases; an orderly way to do this is via "Algorithm 3" from [6]. Unfortunately, this greedy method is not always optimal; in particular, we can get "stuck" in a local minimum. (An example showing this situation is presented in Appendix A.2 under "A greedy strategy".)

We can divide the N_B elements into N_G groups in polynomial time using the following dynamic programming method. Recall that we assumed the elements in p to be in non-increasing order. (This can be ensured by sorting p.) Let $T(i, j, k)$ be the minimum expected number of messages possible by concentrating only on stations $\{i, i + 1, \ldots, j\}$ and partitioning this set of stations into $k \geqslant 1$ paging groups, for $j \geqslant i + k - 1$. Algorithm *Group* wants to find the partition corresponding to $T(1, N_B, N_G)$.

Clearly, $T(i, j, 1)$ can be computed, for all i, j. Iteratively compute $T(i, j, k)$ as follows: from Lemmas 3 and 4, the best way to partition $(i, i + 1, \ldots, j)$ into k sets is to find a station u such that $\{i, i + 1, \ldots, u\}$ is the first set and $\{u + 1, u + 2, \ldots, j\}$ is divided into $k - 1$ sets. In particular, by using the formula for $E(M)$ given in (5) we get

$$T(i, j, m + 1) = \min_{i \leqslant u \leqslant j - m} Z(i, u, j, m),$$

where $Z(i, u, j, m)$ equals

$$T(i, u, 1) + T(u + 1, j, m) + (u - i + 1) \sum_{q=u+1}^{j} p_q .$$

Algorithm *Group* first sorts the N_B probabilities, if required, and precomputes for all $1 \leqslant i, j \leqslant N_B$ the partial sums $\sum_{u=i}^{j} p_u$. It then computes the values $T(i, j, k)$ for all $1 \leqslant i \leqslant N_B - N_G$, $i \leqslant j \leqslant N_B$, and $1 \leqslant k \leqslant N_G$ in $O(N_B^2 N_G + N_B \log N_B)$ time. (Precomputing the $\sum_{u=i}^{j} p_u$'s helps in determining $T(i, j, k + 1)$ iteratively in $O(1)$ time.) The partition corresponding to each $T(i, j, k)$ can be maintained implicitly by storing the corresponding value of u with each $T(i, j, k)$.

4.3. Can the gap be closed?

There is a gap between our upperbound of $O(N_B^2 N_G)$

and the sorting lowerbound (as mentioned below Lemma 4) of $\Omega(N_B \log N_B)$ we have for this problem. Certain optimizations can be done for specific values of N_G; for example, in section 4.1 we looked at the important case when $N_G = 2$.

The following lemma helps in cutting down the computation for general N_G by a constant factor. (The big-oh bound still remains $O(N_B^2 N_G)$.)

Lemma 5. Amongst all possible partitionings of the N_B stations into N_G paging groups that minimize $E(M)$, there is one partitioning into sets B_1, B_2, $\ldots B_{N_G}$ such that $\forall j > 1, k_{j-1} \leq k_j$.

Proof. Lemma 3 implies that there is a configuration that minimizes $E(M)$ such that for any $1 \leq i \leq N_B - 1$, stations i and $i + 1$ are in the same paging set B_j or in consecutive paging sets. Consider such an optimal partitioning. Let i be the station with the smallest probability in set B_{j-1}. It can be verified that we can move station i from B_{j-1} to B_j without increasing $E(M)$ if

$$p_i \cdot (k_j + 1) \leqslant P_{j-1} . \tag{7}$$

If i is the station with smallest probability p_i amongst stations in B_{j-1}, it implies that every station in B_j has an associated probability at most p_i. If in the optimal partitioning $k_{j-1} \geqslant k_j + 1$, then $p_i \cdot (k_{j-1}) \geqslant p_i \cdot (k_j + 1)$. Since $P_{j-1} \geqslant p_i \cdot k_{j-1}$, it follows that $p_i \cdot (k_j + 1) \leqslant P_{j-1}$. From (7), station i can be moved from B_{j-1} to B_j. This argument can be repeated until the desired configuration is attained. $\qquad\square$

5. Conclusions

In this paper we have looked at important issues in searching for a mobile phone. We have looked at the question of queuing delays and derived the result that parallel search may not reduce the delay in searching for a phone if the system is heavily loaded. In particular, our analysis sheds light on important design issues in paging in cellular telephone networks. We find that at high loads, the delay decreases significantly in going from one to two paging groups. When considered along with the number of messages sent, perhaps 2–4 paging groups is a good value; e.g., having four paging groups reduces the expected total number of messages (relative to flooding) from N_B to $0.625 N_B$. The minimum is $0.5 N_B$, achieved with flooding which leads to a high delay. It would be interesting to study the impact of queuing delay on other results in the literature.

We have also presented an efficient method to group stations into paging groups when a probability vector defining the likelihood of finding a phone in any paging area is given, and the goal is to minimize the expected number of messages. Our algorithm *Divide2* optimally and efficiently partitions the states into $N_G = 2$ paging groups, an important case as suggested by our delay analysis. For general N_G, our algorithm *Group* takes worst case time proportional to $O(N_B^2 N_G + N_B \log N_B)$ to determine the optimal partitioning. It would be interesting to close the gap between our known lowerbound of $\Omega(N_B \log N_B)$ and the upperbound of *Group*. Another open issue is to derive an appropriate model for analyzing the effect of queuing delays when the probability vector is given. Some discussion related to these open issues appears in the Appendix.

Acknowledgements

Appendix

Possible future work

In this appendix, we present some facts that shed more light on possible future work.

A.1. Minimizing $E(M)$ for $N_G = 2$

We continue here with the question we posed at the end of section 4.1: How unbalanced in terms of probability can sets B_1 and B_2 be? Can $\Delta E(M)$ be negative if $P_{k-1} > 1/2$? In other words, do we always exit the second loop in Table 3 when the first set has cumulative probability $> 1/2$. The following example shows that we may not.

Consider $p = (5/8, 1/3, 1/240, 1/240, \ldots, 1/240)$, for example. Although $5/8 > 1/2$, the optimal partitioning is: $\{5/8, 1/3\}, \{1/240, \ldots, 1/240\}$.

A.2. Minimizing $E(M)$ for general N_G

In this section, we discuss ideas towards developing a better-than-$O(N_B^2 N_G)$ solution to the problem of grouping base stations into paging groups when the probability vector is given. (This continues the discussion from section 4.3.) We present counter-examples for two intuitive ideas.

The balanced probability idea. The first idea says that there exists a configuration that minimizes $E(M)$ such that "the paging groups have roughly the same cumulative probability". (We need to ensure that Lemma 3 is obeyed.) The following example shows, however, that the probabilities of the paging groups could be fairly different.

Consider the situation with $N_B = 12$ stations and the probability vector $p = (1/3, 1/3, 1/30, \ldots, 1/30)$. The most "balanced" partitioning (in terms of probability) is into paging groups $B_1 = \{1\}$, $B_2 = \{2\}$, and $B_3 = \{3, 4 \ldots, 12\}$, with $P_1 = P_2 = P_3 = 1/3$, and $E(M) = 5$. However, the partitioning into paging groups $B_1 = \{1, 2\}$, $B_2 = \{3, 4, 5, 6, 7\}$, and $B_3 = \{8, 9, 10, 11, 12\}$ has $P_1 = 2/3$, $P_2 = P_3 = 1/6$, and $E(M) = 4.5$.

A greedy strategy. A more intuitive idea (e.g., "Algorithm 3" from [6]) is a simple greedy strategy. That is, along the lines of (6), we can derive a condition under which $E(M)$ does not increase by moving the highest probability station i from set j to set $j - 1$. Such conditions suggest that we might get an optimal partitioning of the stations into sets by locally moving elements across "adjoining" sets.

More precisely, if we start with a partitioning of the N_B stations into N_G sets obeying the property stated before Lemma 3 (that for any $1 \leqslant i \leqslant N_B - 1$, stations i and $i + 1$ are in the same paging set B_j or in consecutive paging sets), and move stations between adjacent sets if $E(M)$ decreases (i.e., greedily), will we reach a configuration that minimizes $E(M)$? If so, we can generate optimal partitionings in $O(N_B N_G)$ time [6].

The following example shows that we may not. For our counter-example, we use $N_B = 23$ stations with $p = (1/3, 49/150, 1/60, 1/60, \ldots, 1/60, 1/150)$. Let $N_G = 3$. The optimal partitioning is into sets $\{1, 2\}$, $\{3, 4, \ldots, 13\}$, $\{14, 15, \ldots 23\}$ with $E(M) = 1096/150$.

If we start with a partition of stations into sets $\{1\}$, $\{2\}$, $\{3, 4, \ldots, 23\}$ and use the greedy method (or "Algorithm 3" from [6]), we will end up with a final partition of stations into sets $B_1 = \{1\}$, $B_2 = \{2, 3\}$, and $B_3 = \{4, 5, \ldots, 23\}$ with $E(M) = 1320/150$, which we know is not optimal. That is, we get "stuck" in a local minimum. The basic problem here is that to get out of the local minimum, we need to make two changes *simultaneously*; i.e., move station 2 to set B_1 and station 4 to set B_2.

We are currently looking at variants of the balanced probability and greedy heuristics to get a better-than-$O(N_B^2 N_G)$ optimal strategy.

A.3. Minimizing delay when queue lengths are known

In this section, we look at the simple case when the queue length at each station is known prior to choosing a paging order, and we want to minimize the delay. This is a first step towards analyzing queuing delay for general non-uniform cases.

When the request to locate the phone arrives, let the queue length for the ith station be q_i. We assume q_i is independent of time; in other words, the time for station i to broadcast a paging message after it receives a request to page is proportional to q_i, independent of when the request is received. Our goal is to group the stations into N_B/k sets of k stations each so that the expected time to locate the phone is minimized. As in section 3.2, the expected delay at any stage is proportional to the maximum queue length amongst the k stations that page in the stage.

Without loss of generality, assume $q_1 \leqslant q_2 \leqslant \cdots \leqslant q_{N_B}$. We show that the algorithm A in which stations j, where $(i - 1)k + 1 \leqslant j \leqslant ik$ are requested to page for the phone in the ith stage, for $1 \leqslant i \leqslant N_B/k$, is optimal.

Lemma 6. When each station is equally likely to find the phone, Algorithm A has minimum expected delay.

Proof. (Sketch) The proof follows directly by convexity arguments. For any algorithm, let r_j, $1 \leq j \leq N_B/k$ be the maximum queue length amongst stations that page in the jth stage. The expected delay is proportional to $\sum_{j=1}^{N_B/k} k/N_B (\sum_{m=1}^{j} r_m)$. Let B be the algorithm that incurs minimum expected delay. It follows that for algorithm B, $r_j \leq r_{j+1}$, $\forall j$. By simple induction, there are at least kj stations with queue lengths at most r_j. The above two facts imply that algorithm B incurs at least as much delay as algorithm A. $\square$

References

[1] B. Awerbuch and D. Peleg, Concurrent online tracking of mobile users, *Proc. 1991 ACM SIGCOMM Conf.*, pp. 221–233.

[2] A. Bar-Noy, I. Kessler and M. Sidi, Mobile users: To update or not to update, Wireless Networks 1 (1995) 175–185.

[3] A. Bar-Noy and I. Kessler, Tracking mobile users in wireless networks, *INFOCOMM '93*, pp. 1232–1239.

[4] S.T.S. Chia, Location registration and paging in a third generation mobile system, BT Tech. J. 9(4) (1991) 61–68.

[5] R.H. Katz, Adaptation and mobility in wireless systems, IEEE Personal Commun. (First Quarter 1994) 6–17.

[6] S. Madhavapeddy, K. Basu and A. Roberts, Adaptive paging algorithms for cellular systems, *Proc. Fifth WINLAB Workshop on Third Generation Wireless Information Networks*, April 1995, pp. 347–361.

[7] K.S. Meir-Hellstern, E. Alonso and D. O'Neil, The use of SS7 and GSM to support high density personal communications, *Proc. Int. Conf. on Communications (ICC)*, 1992.

[8] S. Mohan and R. Jain, Two user location strategies for personal communication services, IEEE Personal Commun. (First Quarter 1994) 42–50.

[9] C. Rose and R. Yates, Ensemble polling strategies for increased paging capacity in mobile communication networks, manuscript.

[10] C. Rose and R. Yates, Minimizing the average cost of paging under delay constraints, Wireless Networks 1 (1995) 211–219.

D.J. Goodman (IEEE M'67-SM'86-F'88) was born in Brooklyn, NY, in 1939. He received the B.S. degree from Rensselaer Polytechnic Institute, Troy, NY, the M.S. degree from New York University, New York, NY and the Ph.D. degree from Imperial College, University of London, London, U.K., all in electrical engineering. Since September 1988, he has been a Professor in the Department of Electrical and Computer Engineering at Rutgers, the State University of New Jersey, Piscataway, NJ. He is also Director of the Rutgers Wireless Information Network Laboratory. Prior to joining Rutgers, he was with AT&T Bell Laboratories as a Department Head in the Communications Systems Research Laboratory. His research has spanned many areas of digital communications, including wireless information networks, digital signal processing, digital coding of speech signals, and speech quality assessment. In 1995 he was a Research Associate at the Program on Information Resources Policy at Harvard University.
E-mail: dgoodman@winlab.rutgers.edu

P. Krishnan (who is called "Krishnan" or "PK") is a member of technical staff at Bell Laboratories. His research interests include network and service management, the development and analysis of algorithms, online and prediction algorithms, prefetching and caching, mobile computing, and data compression. He received his Ph.D in computer science from Brown University in 1995, and his B. Tech in computer science and engineering from the Indian Institute of Technology, Delhi, in 1989.
E-mail: pk@bell-labs.com

Binay Sugla is a distinguished member of technical staff at Bell Laboratories. He obtained his Ph.D. in 1985 from the University of Massachusetts at Amherst. In 1989 he completed CAPER, a visual parallel programming environment used to write and run applications on parallel-distributed networks. For the past few years he has been working in the area of network and service management. As part of this he has built a tool called the Network Flight Simulator to enable real-time simulation and management of large networks and services. Currently, he is working on Internet Management.
E-mail: sugla@bell-labs.com