A NUMERICAL ALGORITHM FOR THE GI/G/k QUEUE

D.E. Everitt* and T. Downs**

* British Telecom, London, England
** University of Queensland, Brisbane, Australia

ABSTRACT

In this paper a numerical algorithm is presented for the transient analysis of the GI/G/k queue. Most of the ideas are explained in terms of the M/G/k queue and their ready extension to the GI/G/k system is demonstrated. The algorithm is easily applied to a range of queueing disciplines and examples of several such disciplines are discussed. Some numerical examples are given with plots of transient behaviour and some approaches to the efficient calculation of steady state distributions are also considered.

1. INTRODUCTION

It is well-known that, in general, multiserver queues are much more difficult to analyse than their single-server counterparts. The only multiserver queues whose stationary distributions are known in an explicit analytic form are the queues M/M/k and GI/M/k (whose solution involves the determination of the root of a certain functional equation). Apart from a few special cases (see for instance [1]) techniques for the analysis of the other types of multiserver queue have taken the form of computational algorithms or approximation techniques. In [2] and [3] computational algorithms were given for the calculation of stationary distributions relating to the M/E_r/k queue and this work was extended to the GI/E_r/k queue in [4] and [5]. The approximation techniques have taken a variety of forms, some involving considerable ingenuity (see for instance [6-10]). All of these methods have been concerned with the determination of stationary distributions; there is little or no literature available on the determination of transient behaviour of multiserver queues.

In this paper we present a numerical algorithm for the transient analysis of the GI/G/k queue. The basic ideas are most readily explained in terms of the M/G/k queue and, as a consequence, sections 2 - 7 are concerned with M/G/k. The method is fairly easily generalised to GI/G/k, as explained in section 8. For clarity, the algorithm is developed in terms of a two-server system; extension to the general multiserver case is quite straightforward. The algorithm is also readily adapted to a wide range of queueing disciplines; several of these are discussed in section 3.

2. DEVELOPMENT OF A NUMERICAL ALGORITHM FOR THE M/G/k QUEUE

As mentioned above, the algorithm will be developed in terms of a two-server system. For the M/G/2 queue, we consider the following model. At epoch n, X_n customers arrive at the queue, with

Pr(X_n = 1) = \lambda \quad \text{and} \quad Pr(X_n = 0) = 1 - \lambda

Just before epoch n, the unfinished work at the first server is denoted by W(1)_n, and similarly W(2)_n denotes the unfinished work at the second server. Thus the virtual waiting time W_n at epoch n is given by

W_n = \min(W(1)_n, W(2)_n).

Arriving customers join a single queue, and are served in order of arrival by the servers. The FIFO queue discipline is entirely equivalent to choosing the server for which W(1)_n is the smallest. If an arriving customer finds only one server free, he will be served by that server, but if he finds both servers free he will be served by the first server. (Various other assumptions are possible here and will be discussed later.)

A customer served by the first server has a service requirement S(1)_n with probability generating function (pgf) B(1)(z); similarly a customer served by the second server has a service requirement S(2)_n with pgf B(2)(z).

The model can therefore be described by the following recurrence relations

\[ \begin{align*}
W(1)_{n+1} &= \left( W(1)_n + S(1)_n \right) \cdot U(W(2)_n - W(1)_n - 1) + \\
W(2)_{n+1} &= \left( W(2)_n + S(2)_n \cdot U(W(1)_n - W(2)_n - 1) - 1 \right)
\end{align*} \]

where

\[ U(x) = \begin{cases} 
1 & x \geq 0 \\
0 & x < 0 
\end{cases} \quad \text{and} \quad (a)^+ = \max(0, a) \]

Define the pgf of the Markov chain \( (W(1)_n, W(2)_n) \) to be

\[ P_n(z_1, z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j \cdot Pr(W(1)_n = i, W(2)_n = j) \]

Also, define the 'partial' generating functions

\[ P_n(1)(z_1, z_2) \quad \text{and} \quad P_n(2)(z_1, z_2) \]

as follows

\[ P_n(1)(z_1, z_2) = \sum_{i \leq j} z_1^i z_2^j \cdot Pr(W(1)_n = i, W(2)_n = j) \]

and

\[ P_n(2)(z_1, z_2) = \sum_{i \geq j} z_1^i z_2^j \cdot Pr(W(1)_n = i, W(2)_n = j) \]

\( P_n(1)(z_1, z_2) \) is therefore the partial generating function corresponding to the set of states.
The generating function form of (1) is then given by
\[ P_{n+1}(z_1, z_2) = \Pi \left[ (1 - \lambda A (z_1) P_n(z_1, z_2)) \right]_{z_1=0, z_2=0}^{\infty} \]
where the \( \Pi \) operator is the z-domain equivalent of the \( (\cdot)^* \) operator; in two dimensions it is defined in the fairly obvious way by
\[ R \{ \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} c_{ij} (z_1 z_2)^i j \} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} c_{ij} (z_1 z_2)^i j \]
The derivation of (2) from (1) may not be obvious to some readers. A direct probabilistic proof is given in the Appendix. Relation (2) is easily seen to the set of states which will be served by the first server. Many other assumptions are also possible, and some of these will now be mentioned.

(i) We can assume that a customer finding both servers free will join the first server with probability \( \pi_1 \) and the second server with probability \( \pi_2 = 1 - \pi_1 \). If we redefine \( P_n^{(1)} \) and \( P_n^{(2)} \) by
\[ P_n^{(1)}(z_1, z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} c_{ij} (z_1 z_2)^i j \cdot \pi_1 \]
and
\[ P_n^{(2)}(z_1, z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} c_{ij} (z_1 z_2)^i j \cdot \pi_2 \]
Then equation (2) will still apply.

(ii) As pointed out by Krishnamoorthi [11, the assumption that a customer will always be served by a free server may not be appropriate in the case of heterogeneous servers where one server is considerably slower than the other, since the customer might be served more quickly if he waits for the faster server rather than beginning service immediately at the slower server. We might therefore assume that a customer chooses the second server only if the waiting time at the first server at the time of the customers arrival is at least \( M \) time units more than that of the second server.

We therefore define
\[ P_n^{(1)}(z_1, z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} c_{ij} (z_1 z_2)^i j \cdot \pi_1 \cdot \pi_2^{i+j} \]
and
\[ P_n^{(2)}(z_1, z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} c_{ij} (z_1 z_2)^i j \cdot \pi_2 \cdot \pi_1^{i+j} \]
and again equation (2) is satisfied.

The splitting of the matrix \( P \) can be represented diagrammatically as follows

\[ P = \begin{bmatrix} m \\ P^{(1)} \\ P^{(2)} \end{bmatrix} \]
We may consider a model where the second server operates only under overload conditions. Therefore, assume we have a threshold \( M \) such that for \( W_1^{(1)} > M \), an arriving customer will always choose the first server. For \( W_1^{(1)} < M \), the first server is considered to be under overload, so the second server begins to serve customers as soon as the overload is relieved. Thus for \( W_1^{(1)} > M \), an arriving customer is served by the second server if \( W_1^{(2)} < W_1^{(1)} \) and by the first server only if \( W_1^{(2)} > W_1^{(1)} \). Diagrammatically this is represented as

\[
P = \begin{bmatrix}
1 & m \\
0 & p(1)
\end{bmatrix}
\]

The overload situation could be treated in a slightly different way. Suppose that an arriving customer will always be served by the first server until \( W_1^{(1)} \) exceeds the threshold \( M \). Once this threshold has been reached, the second server comes into action and remains in action until it becomes free. Once the second server becomes free, it is no longer available for serving customers until \( W_1^{(1)} \) again exceeds the threshold.

We therefore have the following situation.

If \( W_1^{(2)} = 0 \) and \( W_1^{(1)} < M \) choose server \# 1
If \( W_1^{(2)} = 0 \) and \( W_1^{(1)} > M \) choose server \# 2
If \( W_1^{(2)} > 0 \) choose server \# 1 if \( W_1^{(1)} < W_1^{(2)} \)
choose server \# 2 if \( W_1^{(2)} > W_1^{(1)} \).

The splitting of \( P \) is represented by

\[
P = \begin{bmatrix}
1 & m \\
0 & p(1)
\end{bmatrix}
\]

In order to apply this strategy we must be able to measure the state \((W_1^{(1)}, W_1^{(2)})\) of the system at a customer's arrival. However, these quantities often cannot be measured directly. Practical application of such a strategy would therefore require some estimates of the quantities \( W_1^{(1)}, W_1^{(2)} \), based on some directly measurable value, e.g. the number of customers waiting at the queue.

In general, we can assume that a customer arriving to find state \((W_1^{(1)}, W_1^{(2)}) = (i,j)\) will choose the first server with probability \( r_{1j}^{(1)} \) and the second server with probability \( r_{1j}^{(2)} \), \( r_{1j}^{(1)} + r_{1j}^{(2)} = 1, \forall i,j \). Then by defining the 'partial' generating functions \( P_n^{(1)}(z_1,z_2) \) and \( P_n^{(2)}(z_1,z_2) \) as follows

\[
P_n^{(1)}(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j r_{ij}^{(1)} P_n^{(1)}(i,j) \] for \( i_1 = i, i_2 = j \)

\[
P_n^{(2)}(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j r_{ij}^{(2)} P_n^{(1)}(i,j) \] for \( i_2 = i, j_2 = j \)

and

\[
P_n^{(2)}(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j r_{ij}^{(2)} P_n^{(1)}(i,j) \] for \( i_1 = i, j_2 = j \)

\[
eq i_1 = 0 \]

\[
eq j \]

Equation (2) is again satisfied.

4. FURTHER CONSIDERATIONS

4.1 The Distribution of Unfinished Work

If \( U_n^{(1)} \) and \( U_n^{(2)} \) denote the unfinished work at epoch \( n \) at the first and second servers respectively, and \( (U_n^{(1)}, U_n^{(2)}) \) has the generating function \( U_n(z_1,z_2) \), then \( U_n(z_1,z_2) \) is related to \( P_n^{(1)}(z_1,z_2) \) by

\[
U_n(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j P_n^{(1)}(i,j) \]

and (2) can be written

\[
P_{n+1}(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j P_{n+1}(i,j) \]

Thus the algorithm described earlier calculates both the unfinished work and the waiting time distributions.

4.2 The M/G/2 Queue with Identical Servers

Consider the case where both servers are identical, and

\[
U_n^{(1)}(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j P_n^{(1)}(i,j) \]

This is the usual model for a two server queue. If we start with an empty queue, we will always have

\[
P_n^{(1)}(1,j) = P_n^{(1)}(i,1) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} z_1^i z_2^j P_n^{(1)}(i,j) \]

This symmetry can be used to halve the computing time involved in the calculations according to (2).

2. EXAMPLES

Figures 1 to 6 plot some results for the transient and steady state waiting time distributions for the homogeneous server M/G/2 queue calculated from eqn. (2). The examples shown here are for various values of \( \lambda \), and for three different service time distributions, all having a constant expected value of 3.0. These service time distributions are

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Parameters</th>
<th>( \lambda )</th>
<th>( \mu )</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>( \mu = 3 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>SII</td>
<td>( \mu = 3 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>SIII</td>
<td>( \mu = 3 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
Pr[S_n = 1] &= \begin{cases} 1 & i = 1, 2, 3, 4, 5 \\
0 & \text{otherwise} \end{cases} \\
E[S_n] &= \begin{cases} 3 & \text{otherwise} \end{cases} \\
\mu &= \begin{cases} \frac{3(\lambda)\mu - 1}{1 - (\lambda)\mu} & i = 1, 2, \ldots, 10 \\
0 & \text{otherwise} \end{cases}
\end{align*}
\]
\[ E(S) = 3.04 \quad \text{var}(S) = 4.79 \]

Figure 1 plots the expected waiting time \( E(W_n) \) vs. \( n \) for various values of \( \lambda \) and for the service time distribution SII. Figures 2 and 3 plot \( E(W_n) \) vs. \( n \) for the three service time distributions, for \( \lambda = 0.5 \) and 0.6 respectively. The stationary waiting time distributions for these same values are shown in Figures 4 and 5. Figure 6 compares the stationary waiting time distribution of the M/G/2 queue with the corresponding distribution for the M/G/1 queue (with the same utilisation factor \( \rho = 0.9 \) and same service time distribution), illustrating the significant reduction in waiting times for the 2-server queue. Also shown for purposes of comparison is the waiting time distribution calculated as if the 2 servers were independent (i.e., we calculate \( \Pr(W = n) = \Pr(\min(W_1, W_2) = n) \) where \( W_j \) is the waiting time at the \( j \)th (independent) server). Again, the waiting time is greater than that of the M/G/2 queue.
6. NUMERICAL ACCURACY CONSIDERATIONS

Equations such as (2) are ideally suited for accurate numerical calculation on a digital computer since the only operations involved are those of addition and multiplication; thus the subtractive cancellation of approximately equal quantities which is a major cause of numerical inaccuracy in many computations cannot occur.

These comments apply to all the algorithms for the calculation of the time-dependent behaviour of discrete time queuing models, since all these algorithms are basically of the form

$$P_{n+1} = P_n P_n$$

where all the elements of the vectors $P_n$ and matrices $P_n$ are positive. Previous authors (see for example [12]) have noted that their algorithms provide results which are accurate to machine precision.

7. STEADY STATE CALCULATIONS

If the stationary properties of the queue are desired, the above algorithm could be used, where the iterations of the algorithm are continued until steady state is achieved. This will then provide information on both the steady state and transient behaviour of the queue. However, if the transient behaviour is not required, this approach would not be very efficient. Two possible approaches to the calculation of the stationary distribution are given below.

7.1 Approximation Approach

Start with some approximation to the stationary distribution of the queue, and use this as the starting point for the algorithm described earlier. However, the problem of finding an approximation to the stationary distribution is not easy. Reference [8] gives an approximate formula for the average waiting time in the M/G/k queue, but this does not provide enough information on the distribution of waiting times to provide a reasonable starting point for the algorithm. The results of [6,9,10] are more useful, since they give approximations to the waiting time distribution of the M/G/k queue. These therefore give approximations to the distribution of $W = \min(W(1),W(2))$, but we still have the problem of determining the joint distribution of $W(1)$ and $W(2)$ from the marginal distribution of $W$.

Arjas and Lehtonen [7] approximate the $s$-server queue by means of a single server queue. This also leaves us with the problem of determining the joint distribution of $W(1)$ and $W(2)$ from a single marginal distribution, but does give more insight into the operation of the queue.

Thus it seems that there are no simple approximations available as yet which serve as adequate starting points for the above algorithm.

7.2 Matrix Iteration Approach

We could formulate the problem as a matrix iteration problem in the form $P_{n+1} = A P_n$. Thus we must find a constant vector $P$ such that $P = A P$. Matrix iterative techniques can then be used to provide a solution. These techniques are designed to give fast convergence to the solution, and would therefore be expected to converge to the steady state solution much more quickly than the algorithm of eqn. (2).

7.3 Fast Fourier Transform (FFT) Techniques

For long sequences, it is usually more efficient to perform convolutions in the frequency domain (via the FFT) rather than in the time domain. Such improvements in efficiency should be available here, so some economies in computation time could be achieved via the (multidimensional) FFT. This has not been programmed, but the basic form of the algorithm would be the same as for the time domain algorithm. The only problem arises from the fact that we would need to perform the 'splitting' operation of $P$ into $P(1)$ and $P(2)$, and this has no simple frequency-domain equivalent. Thus we would have to continually alternate between the time and frequency domains in our calculations.

7.4 The Method of Phases

While the earlier algorithm based on equation (2) would be expected to be efficient for short-tailed distributions such as those used in the examples, for long-tailed distributions, it may be better to approximate the waiting time distributions by the method of phases as was done by Neuts [13] for the single server queue. Whether or not this is more efficient than the FFT approach has not been investigated.

8. THE GI/G/2 QUEUE

The calculation of the transient properties of the queue GI/G/2 follows much the same as for the M/G/2 queue and so will not be discussed in great detail. Most of that which was earlier said about the M/G/2 queue applies equally well to the GI/G/2 queue.

Consider the heterogeneous GI/G/2 queue where the first server serves a customer in a random time $S(1)$ with pgf $B(1)(z)$ and the second server will serve a customer in $S(2)$ time units, where $S(2)$ has the pgf $B(2)(z)$. The interarrival time $T_{n+1}$ between the arrival of the $n$th and $(n+1)$th customers is assumed to have the pgf $A(z)$.

Also, assume that just before the arrival of the $n$th customer, the unfinished work at the $i$th server (i=1,2) is denoted by $W(i)$. The queue is therefore described by the Markov chain $(W(1),W(2))$. As for the M/G/2 queue, we assume that the $n$th customer is served by the first server if $W(1) < W(2)$ and by the second if $W(2) < W(1)$. (The various other assumptions discussed in the previous section are also possible here.)

The queue evolution is therefore described by the recurrence relations:

$$W_{n+1} = \begin{cases} \left( W(1) + S_n \right) U(2)(W(2) - W(1)) - T_{n+1} \\
W(2) + S_n (W(1) - W(2) - 1) - T_{n+1} \end{cases}$$

where

$$U(x) = \begin{cases} 1 & x > 0 \\
0 & x < 0 \end{cases}$$
Define the pgf \( P_n(z_1,z_2) \) and the partial pgf's \( P_n^{(1)}(z_1,z_2) \) and \( P_n^{(2)}(z_1,z_2) \) as in the previous section. Then in generating function form (6) becomes

\[
P_{n+1}(z_1,z_2) = \mathbb{E}[B^{(1)}(z_1)A^{(-1)}_1P_n^{(1)}(z_1,z_2) + B^{(2)}(z_2)A^{(-1)}_2P_n^{(2)}(z_1,z_2)]
\]

and this can be programmed in a similar fashion to (2).

**APPENDIX**

DIRECT PROBABILISTIC DERIVATION OF THE RECURSIVE FORMULA FOR THE SOLUTION OF THE QUEUE M/G/2

From (1) we have the recursive relations

\[
v^{(1)}[n] = v^{(1)}[n] + X[v^{(1)}[n] - h^{(1)}[n]]
\]

and

\[
v^{(2)}[n] = v^{(2)}[n] + h^{(2)}[n] - v^{(1)}[n] + h^{(1)}[n] - 1
\]

which can be rewritten in a slightly different way

\[
v^{(1)}[n+1] = (v^{(1)}[n] + 1)^+
\]

\[
v^{(2)}[n+1] = (v^{(2)}[n] + 1)^+
\]

Let

\[P_n(i,j) = \Pr[w^{(1)}[n] = i, w^{(2)}[n] = j] \]

\[q_n(i,j) = \Pr[v^{(1)}[n] = i, v^{(2)}[n] = j] \]

\[b^{(1)}[i] = \Pr[s^{(1)}[n] = i] \]

and

\[b^{(2)}[i] = \Pr[s^{(2)}[n] = i] \]

with probability generating functions

\[P_n(z_1,z_2), \quad Q_n(z_1,z_2), \quad B^{(1)}(z_1) \quad \text{and} \quad B^{(2)}(z_2)\]

respectively. Then, from eqn. (A.2) we obviously have

\[P_{n+1}(z_1,z_2) = \mathbb{E}\left[\frac{Q_n(z_1,z_2)}{z_1z_2}\right] \]

Also, from (A.1) by considering all the ways that the state \((v^{(1)}[n] = i, v^{(2)}[n] = j)\) can arise, we have

\[q_n(i,j) = (1-\lambda)P_n(i,j) + \lambda \sum_{k=0}^{\min(i,j)} P_n(k,j)b^{(1)}[i-k] \]

\[+ \lambda \sum_{k=0}^{\min(i-1,j)} P_n(k,i)b^{(2)}[j-k] \]

The first term on the right hand side corresponds to no arrivals occurring at epoch \(n\), the second corresponds to an arrival finding the system in a state where \(w^{(1)}[n] < w^{(2)}[n]\) (i.e. the arrival will be served by the first server), and the third term corresponds to an arrival finding the system in a state where \(w^{(1)}[n] > w^{(2)}[n]\). Multiplying (A.4) by \(z_1z_2\) and summing over all possible values of \(i\) and \(j\) gives

\[Q_n(z_1,z_2) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} q_n(i,j)z_1^iz_2^j = (1-\lambda)P_n(z_1,z_2) + \ldots \]

The second term on the r.h.s. can be rewritten in the form

\[
\frac{\lambda}{1-\lambda} \sum_{k=0}^{\min(i-j,1)} \sum_{j=0}^{\min(i-j,1)} P_n(k,j)b^{(1)}[i-k]
\]

\[
+ \frac{\lambda}{1-\lambda} \sum_{k=0}^{\min(i-j,1)} \sum_{j=0}^{\min(i-j,1)} P_n(i-k,j)b^{(2)}[j-k]
\]

Therefore we have

\[Q_n(z_1,z_2) = (1-\lambda)P_n(z_1,z_2) + \lambda \mathbb{E}\left[\frac{B^{(1)}(z_1)}{z_1}P_n(z_1,z_2)\right] + \lambda \mathbb{E}\left[\frac{B^{(2)}(z_2)}{z_2}P_n(z_1,z_2)\right]
\]

giving, from (B.3)

\[P_{n+1}(z_1,z_2) = \prod_{z_1z_2} \left[1-\lambda + \frac{\lambda}{1-\lambda} \mathbb{E}\left[\frac{B^{(1)}(z_1)}{z_1}P_n(z_1,z_2)\right] + \lambda \mathbb{E}\left[\frac{B^{(2)}(z_2)}{z_2}P_n(z_1,z_2)\right]\right]
\]

as stated in equation (2).

**REFERENCES**

Summary of Questions/Answers

Date: 13 June 1983
Session: 1.1
Paper: 7

Q.1 (J. Augustus)

The dynamic frequency allocation system modelled in your paper requires increased hardware in each cell compared to existing solutions. Does the increased capacity justify the additional expense.

A.1 (D. Everitt)

It must be realized that with radiophone systems, we are dealing with problems rather different than those of conventional telephone networks. In particular, we are limited by having available only a (small) finite number of channels, and this number cannot simply be increased as demand increases as is the case with conventional networks. Therefore, we are presented with the problem of how best to make use of this valuable resource. This is the purpose of the investigation of dynamic channel allocation - it allows us to increase the traffic carrying capacity of a fixed number of radio channels. It would not be considered if there were an ample number of radio channels available for use.

The purpose of this paper was then to quantify the possible increases in traffic capacity due to dynamic channel allocation. This will result in increased control costs and hardware costs; whether or not this is justified by the increased traffic capability depends on the traffic demand and on the geographical layout of the system.

I would also like to comment on the final part of Jim Roberts' reply to the question about his paper. I think that he is probably correct in saying that his equation (7) could be used to evaluate the blocking probabilities for radiophone; however the number of cliques involved is almost as large as the number of cells, so this recursive solution would be almost as difficult to evaluate as summing the state probabilities directly (i.e. a problem of dimensionality). I still feel that the Monte Carlo simulator is really the only useful technique for large networks.
Q.2 (F. Tomé, J. Cunha)

1. Have you any idea about the mean value of the originating traffic per mobile radio subscriber. Which is approximately the rate between this traffic and the traffic per subscriber (residential, business, etc.) of the fix network.

2. Is for mobile radio subscriber usually equal the originating and terminating traffics.

3. Is it possible to tell us approximately the mean holding time of an originating and terminating call in the mobile radio and about its comparison with the behavior of the subscribers of the fix network.

A.2 (D.E. Everitt, N.W. MacFadyen)

The mobile radio scene in the UK is changing so fast and so radically that it is difficult to give useful and meaningful answers - especially to the very important first question.

For the second, mobile-terminating traffic is typically less than 20% of mobile-originating at present. The mean holding-time of a successful call is of the order of 100 seconds, with originating calls casting slightly longer than terminating. This should be compared with some 150 seconds for calls between fixed subscribers in the morning busy-hour.