Dynamic Network Call Distribution with Periodic Updates

Yonatan Levy, Sanja Durinovic-Johri, and Rodolfo A. Milito

AT&T Bell Laboratories, P.O. Box 3030, Holmdel, New Jersey 07733, USA

The ability to effectively distribute calls in the network to multiple customer sites has become one of the major challenges for the Intelligent Network. The general goal is to make effective use of available information to improve the load balancing across the sites and the overall service level customers provide to their own clients. This paper introduces a new approach consisting of two coupled algorithms: a traffic allocation algorithm and a call distribution algorithm. The traffic allocation algorithm dynamically generates the proportion of global traffic to be sent to each site in response to periodic updates of system state. The call distribution algorithm converts this time varying allocation into a sequence of routing decisions. The paper demonstrates that this easy-to-implement approach achieves superior performance even when only a fraction of the offered traffic is controllable, provided that the update interval is not excessively long.

1. INTRODUCTION

This paper addresses one of the challenging problems facing Intelligent Networks (IN): how to effectively use the intelligence in the network to manage customer traffic. Many large corporations, such as airlines, catalog sales, and brokerage firms, rely on telecommunication networks to generate the bulk of their revenues. These corporations typically have multiple sites, and they expect the IN to be able to monitor all these sites and determine the best place to send each call. A contribution to the 13th International Teletraffic Congress[1] discussed the complex nature of this problem, classified different call distribution methods, and proposed a dynamic policy that can be implemented in a centralized network controller. One of the major aspects that makes this problem more complex than the classical problem of routing to parallel queues, which has been widely discussed in the literature (see Weber[2], Houck[3], John[4], and references there), is the distributed nature of the problem which precludes the availability of complete and accurate state information at the decision-making node. While the network controller can keep track of calls in progress at the various sites, it does not have real-time information on changes in site parameters (e.g., number of servers) and on local traffic that is directly routed to the site by private or local networks.

In this paper, we focus on the issue of incomplete information. In the first part (Section 2), we discuss the issue in the context of IN, which leads to the consideration of three practical design problems: (i) how to obtain real-time status information; (ii) where to locate the controller; (iii) how to allocate and distribute the global traffic to the sites. We discuss both the trend of today’s networks and the likely direction of future developments. One observation is that, due to economic considerations as well as the use of other criteria for assigning calls to sites, one should expect neither the availability of perfect information nor the ability to control all the traffic. Hence, in the second part of the paper (Section 3), we provide a precise problem definition and propose a new approach to this problem consisting of two coupled algorithms: an allocation algorithm and a call distribution algorithm. The allocation algorithm determines, based on periodic updates from the sites, the proportional allocation during the update interval, which applies only to the global traffic controlled by the network. While
we do not claim optimality of this algorithm, it has the nice property that it converges to rules known to be optimal for symmetric sites as the update interval approaches either zero (perfect information) or infinity (static rule).

The call distribution algorithm takes as input the periodic allocation and is an extension of the generalized round robin algorithm to a time-varying routing probabilities environment. The goal of this algorithm is to distribute calls to different sites according to a prescribed allocation that can change over time. We prove that it is asymptotically optimal, in the sense that the long-run average deviation of the actual distribution from the intended one converges to zero. It can also be applied to other contexts, for instance, the distribution of tasks to balance the load among several processors.

In Section 4, we use simulation to show the effectiveness of our approach in achieving the goal of using the periodic information to improve the performance and to study the sensitivity of the performance to two critical variables: (i) the update period, i.e. how frequently site updates are sent; and (ii) the fraction of traffic controlled by the algorithm. One of our observations is that controlling about half of the traffic is usually sufficient to compensate for imbalances in the uncontrolled traffic and other local fluctuations.

![Diagram](image)

Figure 1: IN Environment for Traffic Management

2. TRAFFIC MANAGEMENT in INTELLIGENT NETWORKS

The typical environment of multiple customer sites managed by the IN is illustrated in Figure 1. One of the most challenging features of this environment is that it is a highly distributed system which requires centralized decisions. There are two central nodes in the system: one is the customer’s Network Management Center (NMC) and the other is the Network Controller. In a typical current IN, the NMC receives periodic updates from the sites via a private data network, and NMC personnel can interact with the NC using a support system. The NC, on the other hand, actually manages the distribution of calls in the network, based on some stored, mostly static, customer-specific rules. The NC can keep real-time counts of calls in progress in each of the sites. However, these counts are the
'network view' and the NC does not have information on how many calls are actually in queue or in service. In this section we discuss three practical design issues that arise in this environment:

- Collection and processing of site status information;
- Where to locate the decision making node for call distribution;
- How to distribute calls to the various sites. Clearly, the answer to this issue depends on the options chosen for the previous two issues.

The major transport mechanisms for information are: the signaling network, operation support systems, and private data networks. The signaling network is used mainly for call control messages, including the call disconnects that are used by the NC to keep real-time counts of calls in progress at each site. Operation support systems are used to update information in the NC, but today’s systems are too slow to be used for real-time information. Private data networks are being used to provide periodic updates, as frequently as every 15 seconds, of site status to the NMC. As described above, the NC real-time status information on calls in progress does not provide accurate site status. The major missing pieces of data are the number of active agents and the amount of local traffic. There seems to be two possible ways to close the gap in availability of real-time site information: one is to send call-by-call queries from the NC to the NMC and the other is to use the signalling network to update the NC more frequently. Recently announced routing services from AT&T (Intelligent Call Processing) and Sprint use the first method by sending call-by-call routing queries from the NC to a new routing processor which is updated via private data links from the sites. Although the other approach has not been used for call distribution yet, it is used in the AT&T Real-time Network Routing [5]. Of course, another solution can be to use a private network for each customer, but this would be a departure from the basic IN concept. Looking ahead, one of the advantages of using a broad-band network as the backbone for IN services is that real-time information is likely to be available at the network controller.

The next question to consider, given the incomplete information at the NC, is where to locate the decision-making node for call distribution. Until recently, the NC has been the central decision node for all IN services. It acts as a central database where the basic decision tree for the customer is stored. It is a satisfactory solution for static rules, like regional routing, fixed allocation, and also for time-dependent variants of static rules. However, for dynamic rules that need state information, the NC is not an effective decision node, and the NMC has to be considered as an alternative. As mentioned above, this has been the recent trend for the new routing services. Since using the NMC would pose serious reliability problems, new routing processors are introduced that can receive updates via private data networks and call queries from the NC via the signalling network. This could lead to hybrid routing, where some calls are being handled at the NC, and other calls, based on some criteria, are sent to the routing processor for a more dynamic decision.

The question of which call distribution algorithm to use is clearly related to the design choices made on the above two issues. Clearly, for the NC as a decision node, static rules, possibly with updates from the NMC, are a natural choice. One example of such a rule is the use of call allocation algorithms, where the NMC can update the allocation percentages based on site status. However, these updates take a few minutes to execute and take effect, and thus cannot be considered truly dynamic. This mode of operation also requires the NMC personnel to constantly monitor the site updates and/or respond to alarms generated by traffic monitoring systems. Another option is to use the real-time counts of calls in progress in the NC as the basis for a dynamic algorithm. Such algorithms are discussed in Militto et al. [11], and their effectiveness clearly depends on how complete and accurate these call counts and the other site parameters are. For a well-planned operation with few changes in site parameters and with light local traffic, it is a good alternative. However, with the availability of frequent updates from the sites in a separate routing processor, there is a need for a different approach that can effectively use this periodic status information. Such an approach is discussed in the following sections.
3. ALGORITHMS for PERIODIC UPDATES

We begin with a more precise problem definition and some notations. A global stream of calls arrives to a network controller at a rate of $\lambda$. Upon arrival, each call has to be sent to one of $N$ sites. At any time $t$, site $i$ has $s_i(t)$ servers, each serving calls at an exponential rate of $\mu_i$, and $x_i(t)$ is the number of calls at the site (either in service or in queue). Calls may abandon while in queue. In addition to the global arrival stream, each site $i$ can have its own dedicated arrival stream with rate $\lambda_i^0$. and $\lambda^0 = \sum_{i=1}^{N} \lambda_i^0$. $\lambda_i^0$ can also include traffic that is routed by other rules or preferences. In this paper we assume that all arrival rates are known, but the same methods can be used when arrival rates vary and are estimated by the controller.

Every $\tau$ time units, all sites synchronously send a state update to the controller, including $s_i(t)$ and $x_i(t)$. Based on this information, the controller has to decide how to allocate global calls during the next period (until the next update). The general goal is to balance performance across all sites and also to optimize overall performance, where performance can be measured by one of more of the following measures: mean delay, certain delay percentile, fraction of calls lost (due to abandonments and/or blocking), and server utilization. Our approach is to determine, at the beginning of each interval, the traffic allocation that would equalize the expected delays, at the end of the interval, across all sites. It is not readily obvious how well this ‘myopic’ objective would do in achieving the above goal. Our purpose is to show that indeed it is an effective approach.

3.1 THE ALLOCATION ALGORITHM

At each update epoch $t$, the controller calculates target allocations for all the sites to be used until the next update. For $i = 1, 2, ..., N$, let $p_i(t)$ denote the traffic allocation for site $i$ during the interval $(t, t + \tau]$.

At any time $t$, we estimate the expected delay at site $i$ as

$$d_i(t) = \frac{x_i(t) - s_i(t) + 1}{s_i(t)\mu_i}.$$  \hfill (1)

This formula is exact when the service times are exponentially distributed and all servers are busy; otherwise, it is a good approximation, and we use it not for absolute but rather for comparative evaluation between sites. When servers are idle, we still use Eq. (1), which in this case yields a negative value, as a measure of idle capacity.

To simplify notations, we drop in the sequel the dependence on $t$, so that $s_i = s_i(t)$, $x_i = x_i(t)$, etc., and we let $\bar{x}_i(\tau) = E[x_i(t+\tau)]$ denote the expected number of calls in progress at site $i$, and $\bar{d}_i(\tau)$ represent the expected delay in site $i$, both at the end of the update interval. Then

$$\bar{x}_i(\tau) = x_i + p_i \lambda^0 \tau + \lambda_i^0 \tau - s_i \mu_i \tau,$$  \hfill (2)

and based on (1) and (2), we obtain the following expression for $\bar{d}_i(\tau)$:

$$\bar{d}_i(\tau) = \frac{\bar{x}_i(\tau) - s_i + 1}{s_i \mu_i} = d_i + \frac{p_i \lambda^0 \tau}{s_i \mu_i} + \frac{\lambda_i^0 \tau}{s_i \mu_i} - \tau.$$  \hfill (3)

Given that we want to determine the $p_i$'s in such a way that the expected delays across the sites are equal, we have to solve the following system of equations:

$$\bar{d}_i(\tau) = \bar{d}_1(\tau), \quad i = 2, 3, ..., N,$$

$$\sum_{i=1}^{N} p_i = 1.$$  \hfill (4)
The solution to this system of equations is given by:

\[
p_i = \left( \frac{(\lambda + \lambda^0) \tau + \sum_{j=1}^{N} d_j s_j \mu_j}{\sum_{j=1}^{N} \frac{\lambda^0 \tau + \lambda^0_j \mu_j}{s_j \mu_j}} \right) \frac{s_i \mu_i}{\lambda \tau}. \quad (4)
\]

We note that formula (4) also works well when some (or all) sites have idle servers, in which case it balances site occupancies. However, in order to achieve all the equalities in (3), this formula can yield some negative \( p_i \)'s. The following iterative procedure computes (4) in an efficient way and identifies the sites with \( p_i = 0 \); those sites will not receive global calls during the next interval.

**Procedure 1:**

1. Order the sites by \( \hat{d}_i = d_i + \lambda^0_i \tau / s_i \mu_i \); so that \( \hat{d}_1 \leq \hat{d}_2 \leq \cdots \leq \hat{d}_N \).

   Initialize: \( k = 0 \); \( U_0 = V_0 = 0 \).

2. For \( k = 1, 2, \ldots \), calculate recursively

   \[
   V_k = V_{k-1} + \hat{d}_k s_k \mu_k; \quad U_k = U_{k-1} + s_k \mu_k; \quad Z_k = \frac{\lambda \tau + V_k}{U_k}.
   \]

   Compare \( Z_k \) with \( \hat{d}_k \). For the first \( k \) for which \( Z_k \leq \hat{d}_k \), stop and let \( K = k - 1 \).

3. Calculate \( p_i \)'s:

   \[
p_i = \begin{cases} 
   \frac{Z_k - \hat{d}_i}{s_i \mu_i} \frac{s_i \mu_i}{\lambda \tau} & i \leq K \\
   0 & i > K 
   \end{cases}
   \]

**Proposition 1:** Procedure 1 finds the solution to (3) with the additional constraint that \( p_i \geq 0 \) for all \( i \) and with the modification that \( p_i p_j [\hat{d}_i(\tau) - \hat{d}_j(\tau)] = 0 \) for any pair \( i, j \).

The proof follows from the observation that the sign of \( p_i \) in (4), the unconstrained solution, is determined by the sign of \( Z_N - d_i \), where \( Z_N \) is fixed given \( N \).

As the update interval approaches zero, the algorithm approaches a rule that sends each call to the queue with the minimum expected delay, which for the case of symmetric queues simplifies to the Join the Shortest Queue rule, known to be optimal under certain conditions \([21][4]\). As the update interval grows indefinitely, the algorithm simplifies to the Generalized Round Robin routing \([6]\) which is known to be optimal for two queues and any number of symmetric queues. However, the \( p_i \) calculated by the algorithm will not be optimal for very long intervals.

### 3.2 THE CALL DISTRIBUTION ALGORITHM

In this section we face the problem of distributing the global traffic among the \( N \) customer sites according to the proportions \( \{p_i\} \) determined by the traffic allocation algorithm. The index \( n \) will be associated with the \( n^\text{th} \) global arrival. Since the set of allocation probabilities \( \{p_i\} \) can change from one update interval to the next, we use \( \{p_i(\tau_n)\} \) to denote the traffic allocation during the update interval \( \tau_n \) associated with the \( n^\text{th} \) arrival.

For the symmetric, fixed traffic allocation scenario, the round robin is the optimal open loop distribution scheme. This is so, because the round robin meets two criteria: a) the long run fraction of calls allocated to every site is the desired \( 1/N \) fraction of the total traffic; b) the stream seen by each site has the smallest variability among all distribution schemes that meet the first criterion. Hajek \([7]\) obtained the optimal, "most regular sequence" for the fixed traffic allocation, non-symmetric, \( N = 2 \) scenario. Arian and Levy \([8]\) developed the Generalized Round Robin (GRR) algorithm and proved it to be the "most regular sequence" in Hajek's sense for \( N = 2 \), and to be quasi-optimal for \( N > 2 \).
basic idea behind the GRR is to keep track of the actual allocation and to route each arrival to a site whose actual allocation does not exceed its target allocation. Here we further extend the GRR to traffic allocations that change over time.

Key variables in the two algorithms we consider are \( q_i(n) \), \( i = 1, \ldots, N \), \( n = 0, 1, 2, \ldots \). As justified later, \( q_i(n) \) captures the deviation of the number of calls routed to site \( i \) up to the \( n^{th} \) arrival from the intended allocation.

Algorithm A:

i. Initialization:
\[
q_i(0) = 0, \quad i = 1, \ldots, N.
\]

ii. Call routing decision. Let \( a_n \) denote the site to which the \( n^{th} \) arrival is routed.
\[
a_n = \min \{ i : q_i(n-1) \geq 0 \text{ and } p_i(\tau_n) > 0 \}.
\]

iii. State update:
\[
q_i(n) = \begin{cases} 
q_i(n-1) + p_i(\tau_n) & i \neq a_n \\
n_i(n-1) + p_i(\tau_n) - 1 & i = a_n
\end{cases}
\]

Let \( \kappa_i(n) \) denote the number of transactions routed to site \( i \) up to and including the \( n^{th} \) global arrival. Then, \( q_i(n) = \sum_{j=1}^{n} p_i(\tau_j) - \kappa_i(n) \), and the routing decision can be stated as
\[
a_n = \min \{ i : q_i(n-1) \geq 0 \text{ and } p_i(\tau_n) > 0 \}.
\]

This rule reduces to \( a_n = \min \{ i : (n-1)p_i - \kappa_i(n-1) \geq 0 \} \) for time-invariant \( \{ p_i(\cdot) \} \), which is the GRR rule. The GRR tends to penalize the destination with the lowest index, and the ordering affects the sequence. Ariki and Levy[6] obtained the best results by ordering the destinations by decreasing values of the fractions \( p_i \).

In our case, the fractions change over time, and the bias introduced by the ordering becomes even more questionable. Our goal in developing the rule was to obtain a seamless algorithm, in which one routing sequence blends into the next one, even as the underlying allocation changes from measurement interval to measurement interval. Establishing any ordering in the time-varying setting retains the flavor of the sequence of the original time-homogeneous setting, and somehow magnifies its drawbacks. In search of a solution, we note that \( q_i(n) \) measures the deviation of the actual number of calls routed to site \( i \) from the intended allocation of traffic to the same site up to and including the \( n^{th} \) arrival. This interpretation of \( q_i(n) \) suggests the following algorithm, obtained by modifying the decision phase of Algorithm A.

Algorithm B: (Steps i and iii are identical to Algorithm A)

ii. Call routing decision. Recall that \( a_n \) denotes the site to which the \( n^{th} \) arrival is routed.
\[
a_n = \arg\max_{i : p_i(\tau_n) > 0} \{ q_i(n-1) \geq 0 \}.
\]

The following results hold for the two algorithms.

**Proposition 2**: The set of real numbers \( q_i(n) \) satisfy \( \sum_i q_i(n) = 0 \) for every \( n \).

This property follows from the initialization \( q_i(0) = 0 \), and the fact that at every arrival epoch the algorithm adds 1 and subtracts 1 to the sum.

An immediate consequence of Proposition 2 is the following

**Corollary 2**: Algorithms A and B are equivalent for the case \( N = 2 \).

Recall that algorithm A collapses into the GRR algorithm for constant \( \{ p_i \} \), and that the latter is optimal for \( N = 2 \). Therefore, we conclude that algorithm B is optimal for the problem of distributing calls to two sites with fixed traffic allocation.
Proposition 3: For every \( i \) and \( n \), \(-1 \leq q_i(n) \leq N - 1\).

First note that \( q_i(n) = \sum_{j=1}^{n} p_i(\tau_j) - \kappa_i(n) \geq -1 \). This holds because \( q_i(0) = 0 \), and at the \( j^{th} \) arrival the state variable \( q_i(j-1) \) is either incremented by \( p_i(\tau_j) \) or decremented by \( 1 - p_i(\tau_j) \). The latter can happen only if \( q_i(j-1) \geq 0 \). The fact that there is at least one \( q_i(j-1) \geq 0 \) is guaranteed by Proposition 1. The upper bound \( N-1 \) follows immediately from the extreme case scenario: \( q_i(n) = -1, \forall i \).

Corollary 3: For both algorithms, \( \lim_{n \to \infty} \frac{q_i(n)}{n} = 0 \).

The statement is trivial in light of Proposition 3, but the result is meaningful given the interpretation of \( q_i(n) \). The long run average deviation of the actual distribution from the intended one is zero.

4. SENSITIVITY to UPDATE INTERVAL and CONTROL LEVEL

The performance of the algorithm is evaluated by simulation with respect to the update interval and the fraction of traffic controlled by the algorithm. We consider three cases: Cases 1 and 2 are symmetric cases with two and three sites respectively, each site with 100 agents (servers) and the same mean service time of 285 seconds. Case 3 is an asymmetric case with two small sites, (each with 50 agents and a mean service time of 300 sec.), and one big site (200 agents and a mean service time of 277.5 sec.). The arrival rates of 2400 calls/hr in Case 1 and 3600 calls/hr in cases 2 and 3 are chosen to give the same offered load equivalent to 95% agent utilization. We assume exponential service times, exponential interarrival times of the global traffic to the NC and dedicated traffic to each site, and an empirical abandonment distribution with mean of 200 sec.. For each case we look at three different operating scenarios with respect to the distribution of dedicated traffic which is not controlled by our algorithm:

- Scenario 1 is the best case scenario in the sense that all sites have the same utilization due to dedicated traffic. The distribution of this dedicated traffic is thus given by (50%, 50%) in Case 1, (33.33%, 33.33%, 33.34%) in Case 2, and (16%, 16%, 68%) in Case 3.

- Scenarios 2 and 3 represent different degrees of imbalance due to the dedicated traffic stream at each site. The imbalance is achieved by shifting 10% of this traffic from Site 2 to Site 1 for Scenario 1 and 20% for Scenario 2. Note that, for a given scenario, the utilizations for sites 1 and 2 are the same for all cases.

Performance of the algorithm is evaluated in terms of the average queuing delays, referred to as Average Speed of Answer (ASA). Results are evaluated in terms of how well the performance is balanced over all sites and also in terms of total performance, and they demonstrate the viability of algorithms of this type. Figure 2 which illustrates the performance of Case 1 is a typical result for the performance relative to the control level. It shows the ASA for Site 1 (dashed line) and Site 2 (solid line) for each scenario as a function of percentage of traffic controlled by the algorithm for an update interval of 15 seconds. The 0% point is a benchmark that shows the performance of the best static rule with a fixed traffic allocation corresponding to Scenario 1, 2 or 3. For Scenario 1, the ASA at the two sites is practically identical and independent of the control level. On the other hand, the benchmark value for scenarios 2 and 3, respectively, shows a big difference in ASA between the two sites and significant improvement with even 10% of the traffic controlled by the algorithm. While the percentage of control required depends on the scenario, it seems that control level of 50% is sufficient.

Similar results were obtained for three sites (cases 2 and 3). We illustrate these cases in Figure 3 by showing the ASA for each site for the worst case scenario and an update interval of 15 seconds. We observe that for the non-symmetric Case 3, more traffic has to be controlled to achieve the performance balance of the symmetric case. We obtained results exhibiting the same behavior for
Fig. 2: ASA versus Control Level - Two Sites

Fig. 3: ASA versus Control Level - Three Sites
other performance measures.

Cases 1, 2 and 3 are also evaluated with respect to the length of the update interval, and we observe a similar qualitative picture in terms of how well the performance is balanced over all sites. Figure 4, which illustrates the worst case scenario for Case 2, is a typical result for the performance relative to the update interval. It shows the abandonment percentage over all sites for update intervals of 15, 60 and 300 seconds.

![Graph showing abandonment percent versus controlled traffic](image)

Fig. 4: Percent Abandonment versus Control Level and Update Interval

As one would expect, system performance improves as the percentage of controlled traffic increases, and as the update interval decreases. Less obvious is the conclusion that controlling as little as 10% of the traffic may result in dramatic improvement in the performance. As the control level increases, the performance improves but at a decreasing rate, until it levels off at approximately 50%. Another conclusion of practical interest is that good performance may be achieved with relatively long update intervals. However, it is worth noticing that the time scale here depends on the time between the events (arrivals or departures) and variability in the number of agents throughout the day.

5. CONCLUSION

Although other approaches can be developed for call distribution in the IN, we believe that the method presented in this paper achieves most of the potential benefits, while being very simple to implement. The important advantage of dynamic algorithms of this type is that they provide NMC personnel with 'auto-pilot' operation, eliminating the need for constantly monitoring the traffic and trying to balance the load among the sites.
REFERENCES


