DYNAMIC ALGORITHMS FOR DISTRIBUTED QUEUES WITH ABANDONMENTS

Rodolfo A. MILITO, Yonatan LEVY, and Yair ARIAN

AT&T Bell Laboratories
Holmdel, NJ 07728, USA.

In this paper we consider the problem of distributing traffic to multiple parallel queues based on incomplete and possibly inaccurate state information. This problem arises in the context of intelligent networks where large customers rely on the network for traffic distribution to multiple locations. The main contributions of the paper are: (i) classification of approaches to the problem; (ii) a revenue-driven, Markovian decision model which captures the crucial elements of the problem: profit maximization, multiple job classes, blocking, and abandonments; and (iii) demonstration of the superior performance of implementable dynamic policies.

1. INTRODUCTION

The demand for a more efficient handling of traffic in today's increasingly complex telecommunication operations poses a challenge to intelligent networks. Of particular interest is the problem of distributing calls in the network to a number of destinations, based on partial and/or inaccurate information. In this paper, we consider the problem of dynamic distribution of calls from one traffic stream to a number of remote queueing nodes, with the general objective of maximizing some performance measure. The problem can be further generalized to multiple traffic streams, each with its own load and revenue characteristics. We intentionally use the term 'distribution' to distinguish the problem from the classical network 'routing' problem. While the problem can be presented as a routing problem, classical network routing does not allow queueing and is concerned only with blocking of calls. In our case, long delays and abandonments are of major concern. While for the carrier, the call might be considered complete when it is delivered to its destination, the customer cannot collect revenues for a call that abandons before service.

The major aspect that makes the dynamic distribution problem different from switched network routing or traffic allocation to parallel queues is the distributed nature of the system: the state of the various queues is not observable by the decision maker and queue state information can only be conveyed periodically or by request. Hence, decisions have to be based on delayed, incomplete or noisy information. The challenge is to use the capabilities of intelligent networks for monitoring traffic and storing information to better estimate the system state and to improve call distribution.

The main contribution of this paper is a revenue-driven, Markovian decision model which captures the crucial elements of the dynamic distribution problem: profit maximization, multiple classes, blocking and abandonments. To set the scenario for this contribution, we first classify existing and potential schemes by various attributes and identify the class of schemes relevant to intelligent networks. After formulating the revenue-driven model, we present several scenarios that exhibit the performance advantage, as measured by expected profit, of using dynamic schemes over static ones. In particular, the robust performance of simple myopic policies in the face of incomplete and inaccurate information is demonstrated.

2. PROBLEM STATEMENT

A global stream of traffic, consisting of several classes of jobs, arrives at a central decision node and has to be distributed to \( N \) remote nodes. Each remote node \( i \) has \( s_i \) servers and \( q_i \) positions for queueing. (It's possible that \( q_i = 0 \) for some nodes.) Jobs of class \( c \) arrive at a rate \( \lambda_{ci} \) and, if processed, require an average service time of \( \mu_{ci}^{-1} \) (at node \( i \)), and generate an expected revenue \( w_{ci} \). Costs are associated with transmitting and processing of jobs and may depend on the destination and class of a job, and \( z_{ci} \) denotes the cost of sending a class \( c \) job to destination \( i \). Hence, job classes are based on both the job type, which is associated with potential revenues, and the job origin, which can affect the costs. Arrivals that find a full system are blocked, and jobs that wait in queue may abandon. In certain scenarios, the right decision might be to block arrivals from certain classes although queueing slots are available. Abandonments are modeled by assuming that upon arrival to node \( i \), a job from class \( c \) that finds \( k \) jobs in queue \( i \) will abandon with probability
\( \xi(k) \), where \( \xi(\cdot) \) is a non-decreasing function which depends on the queue parameters. While not included in this paper, it is possible to convert an abandonment time distribution to the above abandonment probability function.

A distribution policy has to assign each arrival to one of the destinations based on available estimates of the state of the system and its history. An optimal policy is one that maximizes the long-run expected value of some performance measure. Explicit optimal policies are available only for some very simple systems and are limited to minimizing some linear cost function of jobs in system\(^1\) [2]. Completion rate (or throughput of served jobs) and total profit are more appropriate performance measures. However, non-linear objective functions are needed to capture these performance metrics. It's worthwhile noting that while maximizing profit is equivalent to throughput maximization for the case of a single class with equal costs, this is not generally true.

3. CLASSIFICATION of DISTRIBUTION METHODS

Routing and distribution algorithms can be classified according to various attributes, the most significant of which are:

a. Centralized vs. distributed - Are the decisions made in one central node?

b. Static vs. dynamic - Do the decisions depend on the system state?

c. Perfect vs. imperfect state information, with the latter being characterized further by frequency and delay of updates.

In addition to these attributes, the driving factor behind any optimization algorithm is the performance measures used in the objective function. Completion rates, abandonment and blocking probabilities, and delay distributions are all performance measures of interest for our problem. While keeping the expected delay as low as possible can be argued to be equivalent to minimizing abandonments, there is an obvious trade-off between blocking and abandonments. The situation is further complicated in a multi-class scenario, where performance objectives may be class dependent.

The literature on traffic routing to parallel queues focuses almost exclusively on centralized algorithms with perfect information and on delay (or queue length) minimization. The characteristics of intelligent networks lead naturally to centralized schemes of two types: static allocation schemes and dynamic algorithms with imperfect information. In what follows, we briefly review existing static allocation methods and then take a first step towards the development of a dynamic algorithm that encompasses the above performance measures.

3.1 Static Allocation

Static allocation schemes distribute the traffic in a fixed manner, which can depend on system parameters but not on the system state or history. Periodic updates can be used to modify the parameters. Jobs are distributed either randomly, by sending to destination \( i \) with probability \( p_i \), or in a sequential manner, by using a deterministic sequence of destinations. The optimal random allocation for delay minimization can be formulated as a convex non-linear program, based on the fact that the random splitting of a Poisson process yields Poisson processes. However, Hajek\(^3\) established that optimal sequential routing performs better than random allocation and also provided the optimal sequence for two queues. While the optimal sequence for more than two queues is still an open problem - except for the case of symmetric queues for which 'round robin' is optimal - the construction of a good sequence is discussed in Rosberg\(^4\) and Arian and Levy\(^5\).

3.2 Dynamic Distribution Policies

The dynamic distribution problem can be approached by applying the principles of Markov decision theory. The state of the system can be characterized by \( N \) arrays, corresponding to the \( N \) nodes. The length of each array indicates the number of jobs in the corresponding node, including those in service. Every entry in a given array has two components: the first is the class, and the second is the time elapsed since its arrival to the system. Theoretically, it is possible to formulate a decision problem based on this state space and let all the parameters depend on both class and destination. However, this full state information is not available at the decision node. Instead, only a small subset is available in real time. Furthermore, there is no guarantee that the state information is flawless. Hence, we face the rather difficult problem of controlling a partially observable process\(^6\) [7] [8].

4. MARKOV DECISION FORMULATION

We now describe the revenue-driven approach that integrates the aforementioned performance measures in a single framework and captures the important trade-offs among them. We assume that the number of servers at each node is known and that the number of jobs being
processed/queued at the different nodes is tracked at the decision node. While the quest to capture the essence of the problem motivated the consideration of abandonments, the need to find implementable policies forces us to consider certain model simplifications. In particular, we are going to consider the following “balking on arrival” model. Consider that a class c job is routed to queue i, and that, upon arrival, the queue length is k. Then, the job balks with probability $\xi_c(k)$. Jobs that do not balk on arrival remain in queue until serviced. We also assume that service times may depend on the node but not on the class. (This assumption can be relaxed if the state space is augmented so that job counts are kept for each class/node combination.) This model, jointly with standard Markovian assumptions for the inter-arrival time and service time distributions, results in the tractable formulation to be delineated next.

Following the notation introduced in section 2, we use the uniformized discrete time formulation of the problem, where events include arrivals, departures, and bogus transitions. The expected profit during the next interval between events, given a state $x$ and a decision $a$, is

$$r(x,a) = \sum_{c} \frac{\lambda_c}{\nu} f(x,c,a_c),$$

where

$$f(x,c,a) = \begin{cases} w_c(1-\xi_c(x_c)) - z_{ca} & \text{if } a \neq 0 \\ 0 & \text{if } a = 0 \end{cases},$$

and where $x = (x_1, \cdots, x_N)$, $x_i$ is the queue length at node $i$ (not including jobs in service), and $\nu$ is the total transition rate. The decision $a$ is a vector that maps the class space $C$ into the action space $A = \{0, 1, 2, \ldots, N\}$. Action 0 denotes blocking.

Let $S$ denote the state space and consider a stationary policy $\pi_c(x)$, which maps $S \times C$ into the action space $A$. Under Markovian assumptions, standard results in Markovian Decision Processes (see, e.g., Tijms[10]) guarantee the existence of scalars $g$ and $h_x$, $x \in S$ that satisfy

$$g = r(x, \pi_c(x)) + \sum_{y \in S} p_{xy}(\pi_c(x))h_y - h_x,$$

where $p_{xy}(a)$ denotes the transition probability from state $x$ to state $y$ when decision $a$ is chosen, and $g$ gives the long-run average profit. In order to highlight the impact of abandonments in the model dynamics, we derive an explicit expression of $p_{xy}(a)$. To this effect let $[x]^+ \triangleq \max(0, x)$, and define the arrival and departure operators

$$A_ix \triangleq (x_1, \ldots, x_i+1, \ldots, x_N)$$

$$D_ix \triangleq (x_1, \ldots, x_i-1)^+, \ldots, x_N).$$

Then

$$p_{xy}(a) = \begin{cases} \sum_{(c:a_c=i)} [1-\xi_c(x_c)] \lambda_c / \nu & \text{if } y = A_ix \\ \mu_i \min(x_i, s_i) / \nu & \text{if } y = D_ix \\ 0 & \text{if } y = x \end{cases}$$

The maximum expected profit $g^*$ satisfies

$$g^* = \max_a \left( r(x,a) + \sum_{y \in S} p_{xy}(a)h_y - h_x \right).$$

In spite of the existence of efficient algorithms[10] to obtain an optimal policy and its associated maximum expected profit $g^*$, the dimensionality problem precludes in most real situations the numerical computation of the optimal policy. Hence the interest in suboptimal, implementable schemes[11][12]. For the given formulation of the problem, the optimal myopic policy, defined as

$$a^M_{\text{MYOPIC}}(x) = \arg \max_a f(x,c,a)$$

provides a simple to implement, yet effective dynamic rule, as demonstrated by the performance comparisons shown in Section 6. For a discussion of k-step open-loop feedback policies, which attempt to improve upon the myopic rule, see Milito[12].

5. UPPER BOUND ON PROFIT

While the optimal policy and its performance are difficult to obtain in general, it is possible to derive simple bounds on the maximum profit that an optimal policy can possibly produce. These bounds will be used in the sequel as a yardstick for evaluating the performance of the various policies. For a single revenue class, the bound is obtained by having each queue process the maximum possible throughput. In addition, each job is processed with the minimum cost $z_{\text{min}} = \min_j z_j$ and at the fastest rate $\mu = \max_j \mu_j$. Let $s = \sum_{i} s_i$ be the total number of servers in all the queues, then the upper bound for the single class is

$$UB_1 = \lambda^{\text{max}} (w - z_{\text{min}}).$$
where $\lambda_{\text{max}} = \min(\lambda, s\mu)$. Here again, $\lambda$ is the arrival rate and $1/\mu$ the mean holding time of a call.

The upper bound for the multiple class case is obtained in a similar manner. For the case of two classes, let $\lambda_1$ and $\lambda_2$ be the arrival rates of class 1 and class 2 respectively, and assume that class 1 has higher revenue per job $(w_1 > w_2)$. Then the upper bound on the maximum possible profit is:

$$UB_2 = \lambda_1^{\text{max}}(w_1 - z_{\text{min}}) + \lambda_2^{\text{max}}(w_2 - z_{\text{min}}),$$

where

$$\lambda_1^{\text{max}} = \min(\lambda_1, s\mu), \quad \lambda_2^{\text{max}} = \min(\lambda_2, s\mu - \lambda_1^{\text{max}}).$$

Class 1 is of higher revenue jobs and therefore we first try to complete as many jobs as possible from this class. The spare capacity determines the maximum possible arrival rate of class 2 that can be processed. The cost is taken to be the minimum over all locations.

In general, these bounds can be tightened by allowing class dependent service rates and/or by a more elaborate calculation of the minimum cost. However, this is not necessary for the examples in the next section.

6. PERFORMANCE CHARACTERIZATION

When the dynamic schemes described in the previous section are implemented in a decision node, the variables are replaced by their estimates. This makes the robustness of the schemes to errors in the estimates a crucial issue. In this section, we provide performance results, which were derived from a simulation model, comparing static and dynamic schemes under accurate and inaccurate information. Regarding abandonments, it is important to note that while the Markov decision model is based on the simplifying assumption of 'abandon on arrival', the simulation treats abandonments as they really happen.

The reference scenario simulated is that of a system with three remote nodes, 40 servers per node, and a decision node that distributes jobs according to a desired algorithm. Unless otherwise indicated, the maximum queue length at each node is 50, mean service time - 160 units, and mean abandonment time - 60 units (for all classes and all nodes). In each simulation, there is a single class or two classes that differ in their revenue ($w$).

For each revenue class, there are three classes that differ in their cost structure. Each such class has a primary destination with a least cost of 0.10 units, a secondary destination with a cost of 1.00 unit, and a last resort destination with a cost of 2.00 units. The three cost classes differ in their primary destination (even though their revenue structure is the same). Units of time and cost are arbitrary.

Figures 1 and 2 show the expected profit as a function of the offered load per server for the case of a single revenue class. The two figures differ in the revenue $w$ of the class; in Figure 1, we have a class with $w > 1.0$ ($w = 10$), while in Figure 2, $w$ is comparable to the costs ($w = 2.1$). We see that the dynamic scheme is superior to the static scheme irrespective of the load and the revenue. In Figure 1, the advantage is consistently a little over 10%, while for the low revenue class (Figure 2) the advantage is a lot higher, around 85%. We also plot the upper bound on profit derived in Section 5. We see that even though the upper bound is crude, the myopic scheme comes close to it. The difference from the upper bound is larger for the higher revenue jobs (Figure 1), where it ranges from 2% at an offered load of 0.9 to 7% at an offered load of 1.0. This means that a socially optimal rule would not improve by much over the myopic rule, which is optimal for every individual job.
Parameters: single revenue class, 3 queues
\( w=2.1, z_i=0.1,1.0,2.0, s_i=40, q_i=50, 1/\mu=160 \)
mean abandonment time 60

![Graph showing parameters](image)

Figure 2: Expected profit as a function of offered load - low revenue class.

In Figure 3, we present results on robustness with respect to inaccurate information. The error introduced in this case is in the number of servers. The actual number of servers differs from the information that the decision node has. Since there are 3 queues in the simulation, we plot the maximum percentage error over all the queues. We see that although the static algorithm is quite robust to inaccurate information, the dynamic scheme performs still better than a no-error static scheme, even with 25% error. This shows that the advantage of the dynamic scheme over the static scheme is preserved even with large inaccuracies in parameters at the decision node. Figure 3 includes two cases, one with large mean abandonment time of 160 units, and another with mean abandonment time of 60. The higher the mean abandonment time the more robust the dynamic scheme relative to the static scheme. This is due to the increase in the number of abandonments in the node with fewer servers than the algorithm assumes, which results in more jobs being sent to this node.

![Graph showing robustness](image)

Figure 3: Expected profit as a function of percent error when decision node information on number of servers is inaccurate

Figures 4 and 5 exhibit results for two revenue classes. The expected profit achieved by the dynamic rule over all the jobs maintains similar performance as in the single class case relative to the static allocation. Its distance from the upper bound, however, is larger especially for high loads. In Figure 4, we also show the sensitivity of the myopic rule to the queue size, and the expected profit decreases with small queue capacity. These last two observations point to a shortcoming of the myopic rule, it does not block low-revenue jobs when the system is congested or when only a few queue slots are available. The optimal dynamic policy is more likely to look forward and block low-revenue jobs. In Figure 5, we plot the abandonment probability for each of the classes. It is evident that the higher revenue class (class 1) receives preferential treatment and has less abandonments than the low revenue class (class 2). The algorithm makes different routing decisions based on the ratio of cost to revenue. High revenue jobs are sent to locations with least expected delay (unless the delay advantage is very small), resulting in less abandonments, while the low revenue class is sent to the least cost location, resulting in more abandonments.
In conclusion, we demonstrated through a simulation that the myopic dynamic scheme is substantially superior to the static scheme for a range of offered loads and for a wide range of parameter errors. Similar results were obtained for many other scenarios. Moreover, the potential for future improvement by using a better, or even optimal policy is quite limited as demonstrated by the upper bound.

REFERENCES