This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

A Framework for Tiered Service in MPLS Networks

George N. Rouskas
North Carolina State University

Nikhil Baradwaj
MicroStrategy

Abstract— **Many network operators offer some type of tiered service, in which users may select only from a small set of service levels (tiers). Such a service has the potential to simplify a wide range of core network functions, allowing the providers to scale their operations efficiently. In this work, we provide a theoretical framework for reasoning about and tackling algorithmically the general problem of service tier selection. Drawing upon results from discrete location theory, we formulate the problem as a p-median problem under a new directional distance measure, and we develop efficient algorithms for a number of important variants. Our main finding is that, by appropriately selecting the set of service levels, network providers may realize the benefits of tiered service with only a small sacrifice in network resources.**

I. Introduction

With currently available technology, the data rate of optical links is in the order of 2.5-10 Gbps, while 40 Gbps links are becoming commercially available. In order to utilize efficiently this capacity, network operators aggregate several lower-rate traffic streams onto each link. MPLS standards and related technology facilitate the packaging of traffic into label-switched paths (LSPs), and the tunneling of these LSPs from source to destination along higher capacity links.

In a *continuous-rate* network, users may request any rate of service, and the network must be designed so as to accommodate arbitrary requests. For instance, one user may request an LSP with bandwidth of 98.99 Mbps, while another user may ask for 99.01 Mbps. The network provider then faces the problem of designing mechanisms to distinguish between these two rates and enforce them in a reliable manner, a task that may be extremely difficult (or even impossible) for traffic of finite duration. Given the unpredictability of future bandwidth demands in terms of their size, arrival time, and duration, link capacity across a continuous-rate network may become fragmented, posing significant risks to the ability of the network to achieve an acceptable level of utilization and meet users' QoS. A related challenge is that of LSP resizing, i.e., the need to adjust the bandwidth (and possibly, the path) of an LSP if a user exceeds the current allocation.

In practice, most network operators offer some type of *tiered service*, in which users may select only from a small set of service *tiers* (levels). The main motivation for offering such a service is to simplify a wide range of core functions (including network management and equipment configuration, traffic engineering, service level agreements, billing, and customer support), enabling the providers to scale their operations to hundreds of thousands or millions of customers. For instance, a tiered-service network might assign both users requesting 98.99 Mbps and 99.01 Mbps to two LSPs of the next higher available rate, say, 100 Mbps. In this case, there is no need to handle the two LSPs differently; furthermore, the network operator only needs to supply policing mechanisms for a small set of rates, independent of the number of LSPs. Currently, service tiers are either based on the bandwidth hierarchy of the underlying network infrastructure (e.g., DS-1, DS-3, OC-3, etc.), or are determined in some *ad-hoc* manner (e.g., the various ADSL tiers available through different providers).

This work was supported by the NSF under grant CNS-0434975.

In this paper, we develop a systematic framework for selecting the service tiers optimally, eliminating the need for guesswork and *ad hoc* approaches. Our work is based on the observation that, by mapping a user to the next higher offered service level, a tiered-service network may use more resources that a continuous-rate one to satisfy the same set of requests for service; alternatively, for a given amount of resources, a tiered-service network will be able to accommodate a smaller fraction of user requests than a continuous-rate one. Therefore, we address the issue of optimally selecting the service levels to be offered so as to minimize the additional amount of resources required. We formulate the problem as a variant of the p-median problem under a new distance metric, and we present an efficient optimal algorithm which scales to very large number of users. Our main finding is that a small set of optimal service levels is sufficient to approximate the resource usage of a continuous-rate network. In other words, the benefits of tiered service in terms of simplified network functions can be achieved with only a small sacrifice in network resources.

We also consider a variant of the problem in which all service tiers are multiples of a basic bandwidth unit, and we develop efficient algorithms to select the basic unit and service levels that are jointly optimal. A network operating with such a set of service levels would resemble a TDM network that allocates bandwidth in multiples of a slot. Consequently, many robust network management functions developed for telecommunication networks, including admission control, routing, traffic grooming, etc., could be easily adapted for the tiered-service packet-switched network. We emphasize that this "TDM emulation" only concerns the control and management functions, *not* the data plane operation of the network. For example, while bandwidth is allocated in multiples of the basic unit, LSPs are not limited to using a particular slot. Similarly, unlike TDM networks where an unused slot is wasted, excess bandwidth can be allocated to active LSPs by the scheduling algorithm. Furthermore, the bandwidth unit is not fixed or determined by hardware, as in a TDM network, but, it is configurable and can be optimized for the characteristics of the carried traffic. In addition, the routers provide for free the functionality of a time-slot interchange. Overall, the tiered-service network can have many of the benefits, in terms of control and management, of a TDM network, but without the data plane rigidities of such a network.

A restricted version of the problem we consider here was studied in [7] in the context of reducing the number of states for the analysis of ATM networks, and a heuristic based on simulation annealing was used to select the service levels. Our approach is more general, as we provide a theoretical framework for reasoning about and tackling algorithmically the general problem of service tier selection.

In Section II we formulate the problem of service tier selection as a directional p-median problem, and present an efficient optimal algorithm. In Section III we consider the problem with the additional constraint that all service levels be a multiple of the same bandwidth unit; we then develop a set of algorithms to select both the service levels and the bandwidth unit that are jointly optimal. We present numerical results in Section IV, and we conclude the paper in Section V.

II. Service Tier Optimization

We consider a packet-switched network with n users. Let x_i be the amount of bandwidth requested by user i. The network offers $p \geq 1$ levels (tiers) of service; typically, $p \ll n$. The j-th level of service corresponds to bandwidth z_j, $z_1 < z_2 < \ldots < z_p$. In such a tiered-service network, each user i is mapped to service level z_j such that $z_{j-1} < x_i \leq z_j$. The additional bandwidth $z_j - x_i$ represents the performance penalty associated with the tiered service. Given the set $X = \{x_i\}$ of user requests and the number of service levels p, we are interested in finding the set of service levels $S = \{z_1, \ldots, z_p\}$ which minimizes the performance penalty over all n users; we refer to such set as "optimal." In this section, we show that this problem is equivalent to the p-median problem under a new distance measure, the directional rectilinear distance, and we present an efficient optimal algorithm to solve it. In the following, we use terminology standard in discrete location literature, and refer to bandwidth requests as "demand points" and to service tiers as "supply points".

A. The Traditional p-Median Problem on the Line: PM1

The traditional p-median problem asks us to find, for a given set of n demand points on the real line, the set of p supply points that minimizes the total distance of each demand point to its nearest supply point. Let $d(x_i, z_j)$ be the distance from point x_i to point z_j on the line, according to some distance metric. The decision version of the p-median problem on the line may be formally stated as:

Problem 2.1 (PM1): Given a set $X = \{x_1, x_2, \ldots, x_n\}$ of demand points, an integer p, and a bound B, does there exist a set $S = \{z_1, z_2, \ldots, z_p\}$ of p supply points such that $\sum_{i=1}^{n} \min_{1 \leq j \leq p} \{d(x_i, z_j)\} \leq B$?

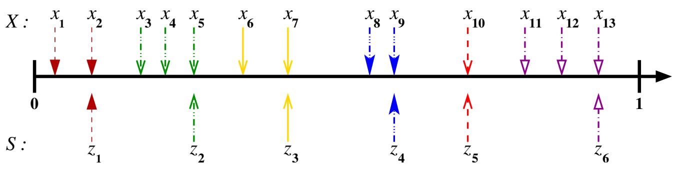

Fig. 1. Sample mapping of demand points x_i to supply points z_j

The choice of distance measure impacts the complexity of the problem as well as the approach needed to find a solution. An $O(np)$ algorithm for PM1 is given in [6], and it is known that the p-median problem is NP-complete in two or more dimensions under either the Euclidean or the rectilinear distance measure [8]. For a comprehensive treatment of location problems, the reader is referred to [5].

B. The Directional p-Median Problem on the Line: DPM1

We now introduce the *directional* rectilinear distance measure. In general, a l-directional, k-dimensional rectilinear metric (with $l \leq k$) defines distance from point $(r_1, \ldots, r_k)$ to $(q_1, \ldots, q_k)$ to be ∞ if $r_i > q_i$ for some $i \in \{1, \ldots, l\}$ and $\sum_{1 \leq i \leq k} |q_i - r_i|$ otherwise. Thus, in a directional p-median problem, a supply point must achieve or exceed the values of the first l coordinates of all the demand points assigned to it. On the real line, this restriction requires that the nearest supply point for a given demand point be located to the right of it, hence, the 1-directional rectilinear distance is:

$$d_{dr}(x_i, z_j) = \begin{cases} z_j - x_i, & z_j \geq x_i \\ \infty, & \text{otherwise} \end{cases} \qquad (1)$$

Let $X = \{x_1, \ldots, x_n\}$ be a set of n demand points on the real line, such that $x_1 \leq x_2 \leq \cdots \leq x_n$, and define the *density* of X to be $\rho_X = \sum_{i=1}^{n} x_i$. A set of supply points $S = \{z_1, \ldots, z_p\}$, $z_1 < z_2 < \cdots < z_p$, $1 \leq p \leq n$, is a *feasible solution* for X if and only if $x_n \leq z_p$. Associated with a feasible solution is an *implied mapping* $X \to S$, where $x_i \to z_j$ if and only if $z_{j-1} < x_i \leq z_j$. Figure 1 shows a sample mapping from a set of 13 demand points onto a set of 6 supply points. Let n_j be the number of demand points mapped to supply point z_j. The directional p-median problem on the real line (which we will refer to as problem DPM1) is:

Problem 2.2 (DPM1): Given a set X of n demand points, $x_1 \leq x_2 \leq \cdots \leq x_n$, find a feasible set S of p supply points, $z_1 < z_2 < \cdots < z_p$, $1 \leq p \leq n$, which minimizes the following objective function:

$$Obj(S) = \sum_{j=1}^{p} (n_j z_j) - \rho_X = \Psi(X, p) - \rho_X \qquad (2)$$

The density ρ_X is the amount of load requested by the original set of demand points, while the term $\Psi(X, p)$ is the load assigned to the users under the tiered service. Hence, $Obj(S)$ is the amount of excess bandwidth needed by the tiered-service network to accommodate the demand set X after mapping it to the supply set S.

The proof of the following lemma is straightforward.

Lemma 2.1: Let X be a set of n demand points. There exists an optimal set of supply points $S = \{z_1, \ldots, z_p\}$, $z_1 < z_2 < \cdots < z_p$, that minimizes the objective function (2) such that $z_j \in X$, for each $j = 1, \ldots, p$.

Define $X_k = \bigcup_{i=1}^{k} \{x_i\}, k = 1, 2, \ldots, n$, as the set with the k smallest demand points in X. Based on Lemma 2.1 and the fact that for a given demand set X the density ρ_X is constant, it is possible to solve DPM1 by using the following dynamic programming algorithm to compute $\Psi(X, p)$ recursively:

$$\Psi(X_1, l) = x_1, \quad l = 1, \ldots, p \tag{3}$$

$$\Psi(X_k, 1) = kx_k, \quad k = 1, \ldots, n \tag{4}$$

$$\Psi(X_k, l+1) = \min_{q=l,\ldots,k-1} \{\Psi(X_q, l) + (k-q)x_k\} \quad l = 1, \ldots, p-1, k = 2, \ldots, n \tag{5}$$

Expression (3) states that if there is only one demand point, it is the optimal supply point. Expression (4) is due to the fact that when $p = 1$, the optimal supply point is equal to the largest demand point. The recursive expression (5) can be explained by noting that the $(l+1)$-th supply point must be equal to the demand point x_k. If the l-th supply point is equal to $x_q, q = l, \ldots, k-1$, the tiered-service load is given by the expression in brackets in the right-hand side of (5), since $k - q$ demand points are mapped to supply point x_q. Taking the minimum over all values of q provides the optimal value.

The running time complexity of the above dynamic programming algorithm is $O(pn^2)$. In the following, we exploit a property of DPM1 to develop an $O(pn)$ optimal algorithm which scales well to large problem instances with demand sets of size n in the order of thousands and beyond.

C. Monge Condition and Totally Monotone Matrices

We can restate DPM1 as a constrained shortest path problem. Let $G = (V, E)$ be a weighted, complete, directed acyclic graph (DAG), with vertex set $V = \{0, 1, \ldots, n\}$. In the DAG representation of DPM1, demand x_i gives rise to vertex i, and we create another vertex 0. Arc weight $w(i, k)$ represents the cost of mapping demand points $x_{i+1}, \ldots, x_k$, to point x_k:

$$w(i,k) = \begin{cases} 0, & k = i+1 \\ (k-i-1)x_k - \sum_{j=i+1}^{k-1} x_j, & k > i+1 \end{cases} \tag{6}$$

As an example, Figure 2 shows the graph for a DPM1 instance with $n = 5$. It is not difficult to prove the following lemma:

Lemma 2.2: Solving an instance of DPM1 with n demand points is equivalent to finding a minimum weight p-link path from vertex 0 to vertex n in the corresponding DAG.

A weighted DAG satisfies the concave Monge condition if

$$w(i,j) + w(i+1, j+1) \leq w(i, j+1) + w(i+1, j) \tag{7}$$

holds for all $0 < i + 1 < j < n$. By substituting (6) into (7), it can be shown that the DAG representing any instance of DPM1 obeys the concave Monge condition.

Consider a matrix M of real elements, and let $I(t)$ denote the index of the leftmost column containing the maximum value in row t of M. Matrix M is said to be *monotone* if

$$t_1 > t_2 \quad \Rightarrow \quad I(t_1) \geq I(t_2), \quad \forall\ t_1, t_2. \tag{8}$$

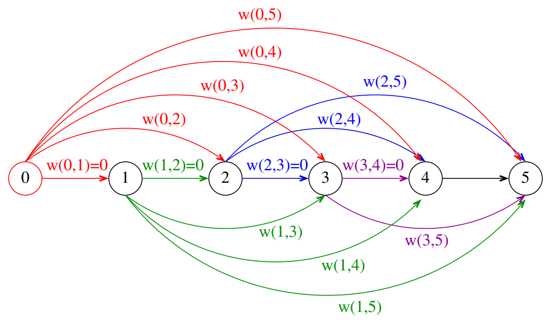

Fig. 2. DAG representation of an instance of the directional p-median problem with $n = 5$

Matrix M is said to be *totally monotone* if all its sub-matrices are monotone [1]. It has been shown [2] that a 2-dimensional Monge array is totally monotone.

The work in [1] presents an algorithm that can find the minimum entry in each column of a totally monotone $n \times m$ matrix, $n \geq m$, in $\Theta(n)$ time. This elegant matrix searching algorithm has many geometric applications, and we show in the next subsection how it can be applied to obtain a faster optimal algorithm for DPM1. For the details of the matrix searching algorithm, the reader is referred to [1], [3].

D. Efficient Dynamic Programming Algorithm for DPM1

To develop a faster algorithm for DPM1, we introduce the following dynamic programming formulation to obtain the optimal value of the objective function $Obj(S)$ in (2):[1]

$$F(1, l) = 0, \quad l = 1, \cdots, p \tag{9}$$

$$F(k, 1) = w(0, k), \quad k = 1, \cdots, n \tag{10}$$

$$F(k, l+1) = \min_{q=l,\ldots,k-1} \{F(q, l) + w(q+1, k)\} \quad l = 1, \ldots, p-1, k = 2, \ldots, n \tag{11}$$

where $w(i, k)$ are the arc weights of the DAG corresponding to this instance of DPM1, as defined in (6). The optimal value for the objective function $Obj(S)$ in (2) is obtained as the value of $F(n, p)$. Expressions (9)-(11) correspond to (3)-(5), respectively, and can be explained in a similar manner.

Our objective is to obtain the value of $F(n, p)$ by computing all the elements of each column l of the matrix defined by F in $O(n)$ time. Note that for $l = 1$, the elements of the first column can be computed in $O(n)$ time from expressions (10) and (6). Therefore, we concentrate on computing expression (11) efficiently. To this end, we introduce a new function $\Gamma(q, k)$:

$$\Gamma(q,k) = F(q, l-1) + w(q+1, k), \quad q, k = 1, \ldots, n \tag{12}$$

We can now see that filling out the l-th column defined by matrix F, i.e., computing the n elements $F(k, l+1), k = 1, \cdots, n$, from expression (11) is equivalent to finding the minimum elements in each column of the $n \times n$ matrix defined by function $\Gamma(q, k)$. Also, $\Gamma(q, k)$ depends on the values of the elements of the $(l-1)$-th column of the matrix defined by F, which have already been calculated.

[1]In contrast, recall that the dynamic programming formulation (3)-(5) was used to compute the optimal value of the term $\Psi(X, p)$ in (2).

We now show that the function $\Gamma(q,k)$ obeys the concave Monge condition. From (7) we know that

$$w(q+1,k)+w(q+2,k+1) \leq w(q+1,k+1)+w(q+2,k) \quad (13)$$

If we add the term $F(q,l-1)+F(q+1,l-1)$ to both sides of (13) and use the definition of $\Gamma(q,k)$ in (12), we get:

$$\Gamma(q,k)+\Gamma(q+1,k+1) \leq \Gamma(q,k+1)+\Gamma(q+1,j) \quad (14)$$

Since $\Gamma(q,k)$ obeys the concave Monge condition, the matrix represented by $\Gamma(q,k)$ is totally monotone [2].

Based on the above observations, in order to solve the dynamic programming algorithm (9)-(11) we proceed by filling the $n \times p$ matrix defined by function $F(k,l)$ one column at a time. The first column ($l=1$) is filled in $O(n)$ time using expression (10). In order to compute the n elements of the l-th column, $l=2,\ldots,p$, we use expression (12) to form an $n \times n$ totally monotone matrix containing $\Gamma(q,k)$ values that depend on the values of the $(l-1)$-th column of the matrix F. The minimum elements in each column of the $n \times n$ matrix Γ are the n elements needed to fill the l-th column of matrix F. These elements can be obtained in $O(n)$ time using the algorithm in [1] we discussed in the previous section. Hence, the time to fill all p columns of matrix F, i.e., the time to find the optimal value for the objective function (2), is $O(np)$.

However, there remains one important issue that we need to address. The $O(n)$ algorithm in [1] assumes that the totally monotone matrix Γ is provided as input, since building this matrix would take time $O(n^2)$. In our case, the matrix Γ is not provided but has to be built anew for computing each column of matrix F. Rather than actually building the matrix in time $O(n^2)$, we now show how to evaluate the value of each element $\Gamma(q,k)$ in constant time. Whenever the algorithm in [1] needs to use the value of some element $\Gamma(q,k)$, rather than accessing the value from memory, we compute its value in constant time. By replacing one constant-time operation (memory access) with another (computing the value), we ensure that the algorithm runs in $O(n)$ time.

From (12) we see that element $\Gamma(q,k)$ depends on the values of $F(q,l-1)$ and $w(q+1,k)$. The value of $F(q,l-1)$ is already computed and hence can be accessed in constant time. In order to evaluate $w(q+1,k)$ in constant time, we perform the following preprocessing operation before starting to solve the dynamic programming algorithm. Define, for $q=1,\ldots,n$, $A(q)=\sum_{i=1}^{q} x_i$. It takes time $O(n)$ to compute and store the values $A(q)$ for all q. Once this is done, one can compute the value of $w(q,k)$ in constant time using the expression:

$$w(q,k) = -A(k)+A(q-1)+(k-q+1)x_k, \quad q \leq k \quad (15)$$

Hence the value of $\Gamma(q,k)$ can be calculated in constant time for any two values q and k.

III. Constrained Service Tier Optimization

In this section we consider a variant of the directional p-median problem in which we impose the constraint that all supply points be multiples of the same unit r. Figure 3 shows a sample mapping from a set X of 13 demand points onto a set S of 6 supply points, under this constraint. The set X is identical to the one in Figure 1 which shows a sample mapping under DPM1. The main difference is that with the new constraint, the supply points are all multiples of the same unit r.

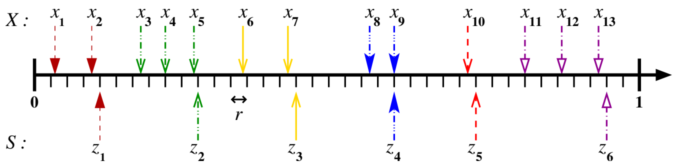

Fig. 3. Sample mapping of demand points x_i to supply points z_j that are multiples of a basic unit r

In the context of MPLS LSPs, parameter r represents the unit bandwidth, i.e., the smallest quantity in which bandwidth may be allocated. Therefore, this problem variant has important applications to next-generation SONET networks in which it is possible to use virtual concatenation to allocate bandwidth flexibly in any multiple of 64 Kbps [4]. It is also applicable to packet-switched technologies (e.g., 1 or 10 GE links) which may benefit from TDM emulation, as we discussed earlier.

This constrained directional p-median problem, which we will refer to as CDPM1, can be expressed as follows (recall that n_j is the number of demand points mapped to supply point z_j, $\rho_X = \sum_{i=1}^{n} x_i$, and $Obj(S)$ is the objective function (2) of the DPM1 problem):

Problem 3.1 (CDPM1): Given a set X of n demand points, $x_1 \leq x_2 \leq \cdots \leq x_n$, and a constant C, find a real r and a feasible set S of p supply points, $z_1 < z_2 < \cdots < z_p$, $1 \leq p \leq n$, so as to minimize the objective function:

$$CObj(S) = \sum_{j=1}^{p}(n_j z_j) - \rho_X + \frac{C}{r} = Obj(S) + \frac{C}{r} \quad (16)$$

under the constraints: $z_j = rk_j$, k_j : integer, $j=1,\ldots,p$.

The term $Obj(S)$ in (16) represents the excess bandwidth penalty, as before. However, the objective function for CDPM1 includes the additional term $\frac{C}{r}$, where C is some constant related to the operation of the system, as we explain shortly. The presence of a term which is a monotonically decreasing function of r in (16) is necessary, since without it CDPM1 reduces to DPM1: if nothing prevents r from being very small, then the optimal is obtained for $r=1$ bps as the solution to DPM1 which minimizes the excess bandwidth penalty.

More importantly, the term $\frac{C}{r}$ is of practical importance as it captures the overhead associated with making the unit r of bandwidth allocation small. To illustrate, let us make the simplifying assumption that all users request and receive the basic rate of r bits/sec. After serving a user, the system incurs some overhead due to the bookkeeping operations, memory lookups, etc., required before it can switch to serving another user. Let α denote the amount of time required to switch between users, expressed as the number of bits that could be transmitted during this time at the given service rate. Therefore, the quantity $\frac{\alpha}{r}$ represents the amount of overhead operations relative to the bandwidth unit. This relative overhead, which increases as the unit of bandwidth decreases, is similar in principle to the "cell tax" incurred in carrying IP traffic over ATM networks due to the relatively large fraction of header (i.e., overhead) bits to data bits. In the objective function (16) we use the term $\frac{C}{r}$ where $C = c\alpha$ and c is a constant which ensures that the two terms in the rightmost side of (16) are expressed in the same units.

We note that the term $Obj(S)$ in (16) requires that the unit r be small so as to minimize the excess bandwidth. However, making r small would increase the term $\frac{C}{r}$ which represents the bandwidth wasted due to overhead operations. Therefore, the objective of CDPM1 is to determine the value of r so as to strike a balance between these two conflicting objectives.

A. Optimal Solution to CDPM1 for Fixed r

As defined, the objective of CDPM1 is to find jointly optimal values for the basic bandwidth unit r (a real number) and the p supply points. However, let us consider for a moment the special case where the value of r is fixed and not subject to optimization; as we shall see shortly, the algorithm for this problem is useful in tackling the general one. In this case, the term $\frac{C}{r}$ in (16) is constant and does not affect the minimization. Hence, the objective function is identical to that of the DPM1 problem.

Consider an instance of CDPM1 in which the value of the basic bandwidth unit is fixed at $r = r_0$; that is, the p supply points can only take the values $kr_0, k = 1, \ldots, K$. Let $U = \{u_1, \ldots, u_K\}$ be the set of candidate values for the p supply points, $u_k = kr_0$; in Figure 3, these candidate values are represented by the ticks below the horizontal line. Integer K corresponds to the largest possible multiple of r_0, i.e., $K = \lceil \frac{x_n}{r_0} \rceil$, where x_n is the largest demand point.

This version of CDPM1 can be represented by a DAG similar to the one in Figure 2. In this case, the DAG has $K+1$, rather than $n+1$, vertices: vertex 0 and the K vertices corresponding to the K candidate values for the supply points (recall that in DPM1, the candidate supply points are the n demand points). Similarly, the arc weight $w(i,k)$ in this DAG represents the cost of mapping the demand points with values between candidate supply points u_i and u_k to u_k:

$$w(i,k) \quad = \sum_{u_i < x_j \leq u_k} (u_k - x_j) \tag{17}$$

It is also not difficult to verify that these weights satisfy the concave Monge condition (7) for all $0 < i+1 < j < K$.

Since this version of CDPM1 has the same objective function as DPM1 and can be represented by a DAG whose weights satisfy the concave Monge condition, we can solve it optimally using the dynamic programming algorithm in Section II-D. Note that the algorithm will run in $O(pK)$, not $O(pn)$, time, as it has to consider K candidates for the p supply points.

B. The Behavior Of The CDPM1 Objective Function

To obtain insight into how the additional parameter r affects the optimization, let us investigate the behavior of the objective function $CObj$ in (16) as we vary r. In Figure 4 we plot the objective function against the value of r for an instance of CDPM1 with $n = 1000$ demand points, $p = 10$ supply points, and $C = 0.01$, with the set of n demand points generated from a uniform distribution in (0,1). We varied the value of the basic unit r in increments of $\delta_r = 10^{-5}$ across the range shown in the figure. For each (fixed) value of r we obtained the optimal supply points in the manner we described in the previous subsection, from which we evaluated the objective function (16), including the term $\frac{C}{r}$.

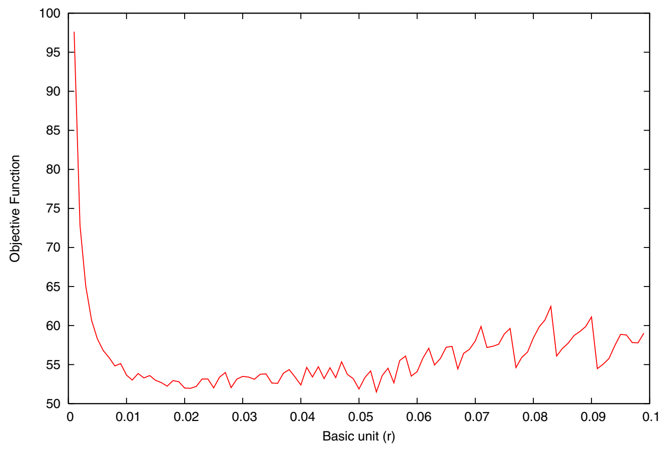

Fig. 4. Objective function value against r, $n = 1000$, $p = 10$, $C = 0.01$, demand points generated from a uniform distribution in $(0, 1)$

The behavior exhibited in Figure 4 is representative of the CDPM1 instances we have studied. At low values of r, the term $\frac{C}{r}$ representing the overhead cost dominates, resulting in large overall values. As r increases, there is an initial period of rapid decrease in the objective function as the term representing the excess bandwidth penalty starts to become important. Following this initial decrease, the curve settles into a seesaw pattern. The high and low points along this pattern depend on the values of the multiples of r relative to the demand points: when multiples of r are aligned close to demand points, there is little bandwidth penalty for mapping these demand points to supply points that are multiples or r, hence the objective function has a lower value; the opposite is true when there is a mismatch between multiples or r and demand points. We also note that as the value of r increases further, the curve trends upwards. This behavior is due to two factors that come into play when r becomes large: the excess bandwidth term in (16) starts to dominate, and at the same time this term increases in value as large values of r are too coarse to minimize the excess bandwidth.

It is clear from Figure 4 that the objective function is non-convex and includes several troughs at irregular intervals. This non-convex nature makes standard optimization techniques (e.g., steepest descent methods) impractical, as it is very easy to get trapped in a local minima. We now describe an exhaustive search approach for identifying the value of r and the supply points that minimize the objective function, and in the next section we develop a suite of heuristics that trade solution quality for running time.

We first observe that for a CDPM1 instance with p supply points and x_n the largest demand point, the largest value that

r may take under the constraint that all p supply points be an integer multiple of r is $r_{max} = \frac{x_n}{p}$. Hence, the optimal value of r lies in the interval $(0, r_{max}]$. Let δ_r be a small increment value, and consider the set $R = \{r_m = m\delta_r \leq r_{max}, m = 1, 2, \ldots\}$. For each (fixed) $r_m \in R$, we use the approach in Section III-A to obtain the optimal supply points, and evaluate the objective function (16). The optimal solution to the CDPM1 problem is obtained as the value of r_m and the corresponding supply points which produce the smallest value for the objective function.

In order to determine the running time complexity of this exhaustive search algorithm, let $L = \lfloor \frac{r_{max}}{\delta_r} \rfloor$ be the size of set R (i.e., the number of candidate values of r to be considered), and $K_m = \frac{x_n}{r_m}$ be the number of candidate supply points when the value of $r = r_m$. The dynamic programming algorithm will be run L times, and during the m-th iteration, i.e., when $r = r_m$, the algorithm will take $O(pK_m)$ time. Since

$$\sum_{m=1}^{L} pK_m = \frac{px_n}{\delta_r} \sum_{m=1}^{L} \frac{1}{m} = \frac{px_n}{\delta_r}(\ln L + \gamma) \qquad (18)$$

where $\gamma = 0.577\ldots$, is Euler's constant, the complexity of the algorithm is $O(\frac{px_n}{\delta_r} \ln(\frac{x_n}{p\delta_r}))$.

As we can see, the complexity of the exhaustive search depends critically on the value of the increment δ_r which determines the granularity of the search. With finer granularity (i.e., smaller δ_r), the accuracy of the algorithm increases, but its complexity also increases dramatically; the opposite is true when δ_r becomes larger and the granularity coarser. We also note that the time complexity is independent of the number n of demand points, and depends only on the largest demand x_n. In the applications we consider, the input value x_n is bounded above by the bandwidth available on the highest capacity link in the network. To get a sense of the values involved in expression (18), consider a network with 10 Gbps links. A reasonable value for the bandwidth increment is $\delta_r = 64$ Kbps. Assuming that the largest demand can be equal to the capacity of a link, we have that $\frac{x_n}{\delta_r} \approx 10^6$, which demonstrates that the exhaustive search is taxing in terms of both computational and memory requirements.

C. Optimization Heuristics

We now present a set of heuristics for the CDPM1 problem. Each heuristic trades solution quality for speed by using its own approach to reduce the size of the space of candidate values for r and/or the supply points that it considers.

1) Demand Driven Heuristic (DDH): Recall that, for each candidate value r_m of r, the exhaustive search algorithm considers all the $K_m = \frac{x_n}{r_m}$ multiples of r_m as the set of potential supply points, where K_m can be much larger than the number n of demand points. The intuition behind this heuristic is that the optimal supply points are more likely to be located just above a demand point, since otherwise there would be a larger penalty in terms of excess bandwidth. Therefore, the heuristic only considers the n multiples of r_m that are located immediately to the right of (or coincide with) the n demand points. In other words, the set U of candidate values for the p supply points is $U = \{u_i = r_m \times \lceil \frac{x_i}{r_m} \rceil, i = 1, \ldots, n\}$. Since there have to be n different candidate supply points, the range of values for r is in the interval $(0, \frac{x_n}{n} = r_{max}]$. Using n instead of K_m and the new value for r_{max} in expression (18), we find that the running time complexity of the DDH heuristic is $O(\frac{px_n}{\delta_r})$, which represents an improvement over the exhaustive algorithm, especially for small values of δ_r which allow for a finer granularity search.

2) Supply Driven Heuristics: Both the DDH and the exhaustive search algorithms apply the dynamic programming algorithm in Section II-D for each candidate value for parameter r. The two heuristics we present in this section are based on the assumption that the optimal supply points for CDPM1 are likely to be close to the optimal supply points for the corresponding unconstrained DPM1 problem with the same demand set. Therefore, each heuristic initially runs the dynamic programming algorithm for the corresponding DPM1 problem, and computes the optimal set $S^{DPM1} = \{z_1^{DPM1}, \ldots, z_p^{DPM1}\}$ of supply points for that problem. This step takes time $O(pn)$, and this dynamic programming algorithm is not used again by the heuristics.

The first algorithm, which we call the *unidirectional supply driven heuristic (USDH)*, sets the i-th supply point for a given candidate value r_m of r to the smallest multiple of r_m that is greater than or equal to supply point z_i^{DPM1}. In other words, the set S_m of supply points for candidate r_m is defined as $S_m = \{\lceil \frac{z_i^{DPM1}}{r_m} \rceil r_m, i = 1, \ldots, p\}$. The heuristic returns the value r_m and corresponding set S_m which result in the minimum value for the objective function (16).

The second algorithm is called the *bidirectional supply driven heuristic (BSDH)*, and computes a set of $2p$ possible values for the supply points for each candidate value r_m. The first set of p values is identical to the set S_m used by the USDH algorithm above. In addition, this heuristic considers the set S'_m consisting of the p largest multiples of r_m that are less than the corresponding supply points z_i^{DPM1}, i.e., $S_m = \{\lfloor \frac{z_i^{DPM1}}{r_m} \rfloor r_m, i = 1, \ldots, p\}$. The $2p$ elements of these two sets collectively become the candidates for being one of the p supply points when the value of $r = r_m$. We use the dynamic programming algorithm in Section III-A to select the optimal set of p supply points from the set $S_m \bigcup S'_m$; it is easy to see that the dynamic programming algorithm works even when the set of the candidate supply points is a proper subset of the set of all integer multiples of r_m. As with USDH, the heuristic returns the value r_m and corresponding p supply points that minimize (16).

We expect the BSDH heuristic to perform better than USDH since it considers a larger number of candidate supply points; this improved performance, however, is at the expense of having to run the dynamic programming algorithm on a set of $2p$ points, which takes time $O(p^2)$.

3) The Power of Two Heuristic (PTH): This heuristic simply selects the set of p supply points as the set of the p consecutive powers of two such that the largest element

TABLE I
FORMULAE FOR THE PDF AND CDF OF THE INPUT DISTRIBUTIONS

Distribution	pdf	cdf	Domain
Uniform	1	x	$0 \le x \le 1$
Increasing	$2x$	x^2	$0 \le x \le 1$
Decreasing	$-2x+2$	$-x^2+2x$	$0 \le x \le 1$
Triangle	$4x$	$2x^2$	$0 \le x < 0.5$
	$-4x+4$	$-2x^2+4x-1$	$0.5 \le x \le 1$
	4/9	$4x/9$	$0 \le x < 0.25$
Unimodal	6	$6x-25/18$	$0.25 \le x < 0.35$
	4/9	$4x/9+5/9$	$0.35 \le x \le 1$
	1/4	$x/4$	$0 \le x < 0.25$
	4	$4x-15/16$	$0.25 \le x < 0.35$
Bimodal	1/4	$x/4+3/8$	$0.35 \le x < 0.65$
	1	$4x-33/16$	$0.65 \le x < 0.75$
	1/4	$x/4+3/4$	$0.75 \le x \le 1$

in the set is the smallest power of two that is larger than or equal to the largest demand point x_n; in other words, $S = \{2^{q+1}, 2^{q+2}, \ldots, 2^{q+p} \mid 2^{q+p-1} < x_n \le 2^{q+p}\}$. This solution is consistent with CDPM1 in that it consists of supply points all of which are a multiple of a basic unit, in this case 2^{q+1}. However, as we shall see shortly, the excess bandwidth penalty for this solution can be quite high compared to the other algorithms. We consider this solution here as a baseline case as it is similar in spirit to approaches that assign packet flows in classes (e.g., as in [9]) whose boundaries are defined by powers of two.

IV. NUMERICAL RESULTS

We now present simulation results to investigate the relative performance of the various algorithms we presented and to determine their effect on the operation of a network. The demand sets X of the problem instances we consider throughout this section were generated from one of six distributions whose pdf and cdf are listed in Table IV. Note that in general, bandwidth demands will be in the range $(0, B]$, where B is the link capacity. However, in order to obtain results that are independent of the link capacity, we assume that all demands are normalized with respect to B; thus, the domain of the pdf and cdf of all distributions in Table IV is [0,1]. Also, based on our discussion in Section III-B and the fact that the largest demand point $x_n \le 1$, we used an increment value $\delta_r = 10^{-5}$ whenever applicable. Due to space constraints, we present only a small set of results here; please refer to [3] for a comprehensive performance study.

Algorithm comparison. Let us first investigate the relative performance of the various algorithms for CDPM1 with respect to the objective function (16). Figure 5 plots the value of the objective function against the value of r for the stated problem instance, and for four CDPM1 algorithms: DDH, BSDH, USDH, and PTH. We also show the optimal value for the corresponding DPM1 problem; this value does not include the overhead term $\frac{C}{r}$ of (16), hence it serves as a lower bound for the CDPM1 algorithms. Note that the DPM1 and PTH solutions do not take parameter r into account, hence they are shown as horizontal lines in the figure. We see that PTH performs much worse than all other algorithms, confirming our earlier observation that using powers of two to define classes of traffic is not an efficient approach; this result is representative of the behavior of PTH, hence, we do not consider this heuristic in the remainder of this section. We also observe that the three algorithms DDH, BSDH, and USDH perform close to the lower bound.

For the results shown in Figures 6 and 7, we have considered thirty problem instances with $n = 100$, $p = 5$, and $C = 0.05$, generated from the increasing and triangle distributions, respectively. The figures plot the objective function value returned by each of four algorithms, DDH, BSDH, USDH, and DPM1, for each problem instance; again, the DPM1 solution provides a lower bound for the other three algorithms. The graphs show that, except for a few instances, all three CDPM1 algorithms are close to the lower bound. Of the three algorithms, DDH produces the lowest objective function values, followed closely by BSDH. The objective function values returned by USDH are generally higher than those of the DDH and BSDH heuristics, but USDH has a much faster running time. Hence, these results indicate that there is a tradeoff between quality of solution and running time complexity of the algorithms.

Bandwidth penalty due to tiered service. Let us now turn our attention to determining the penalty in terms of excess resources needed due to tiered service. Given a demand set X, a continuous-rate link will use an amount of bandwidth equal to $\rho_X = \sum_i x_i$ to satisfy all the demands in X. A link of a tiered-service network, on the other hand, will in general use more bandwidth, as each demand x_i will be mapped to the next offered level of service (i.e., supply point). For a network with service levels obtained through the DPM1 (respectively, CDPM1) algorithm, the amount of bandwidth used is given by the objective function (2) (respectively, the objective function (16) after subtracting the term $\frac{C}{r}$. In our study, we use the *normalized bandwidth requirement* metric, defined as the ratio of the amount of bandwidth used by a tiered-service network to the amount of bandwidth ρ_X used by a continuous network, to characterize the bandwidth penalty incurred by a tiered-service network.

Figures 8 and 9 plot this metric against the number p of service levels offered by the network. Each point in these curves is the average over 30 different problem instances generated by a uniform distribution; similar results were obtained for all other distributions shown in Table IV and can be found in [3].

Figure 8 presents results for two tiered service scenarios: one in which the service levels are obtained from the DPM1 algorithm, and one in which they are obtained from the DDH algorithm; DDH is selected as a representative algorithm for the CDPM1 problem in which the service levels are all multiples of a basic bandwidth unit r. As we can see, the curve for DDH is above the one for DPM1 This result is expected, since the (optimal) DPM1 algorithm is only concerned with minimizing the excess bandwidth due to tiered service, while the DDH algorithm also has to take into account the constraint that all service levels be multiples of a basic unit. However, the additional penalty due to the constraint imposed by the CDPM1 problem is relatively small; we have observed

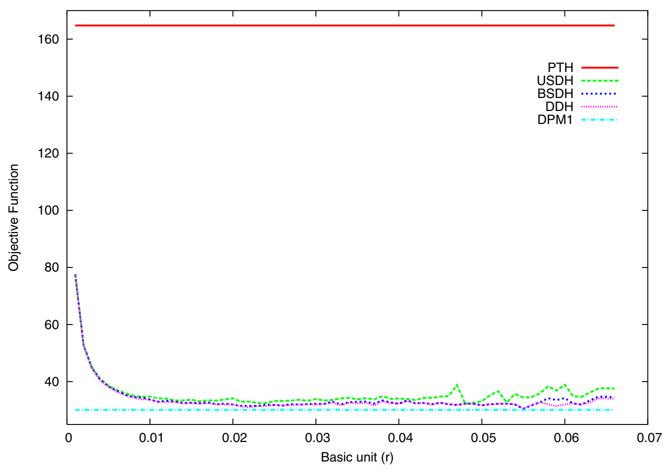

Fig. 5. Objective function value against r, $n = 1000$, $p = 15$, $C = 0.05$, triangle distribution

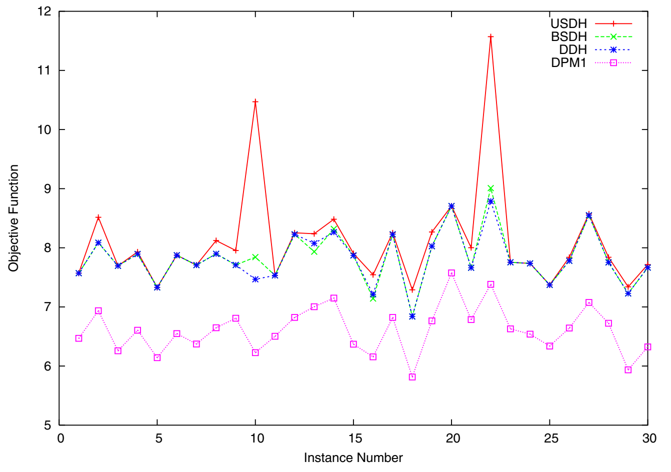

Fig. 6. Objective function value returned by the algorithms, $n = 100$, $p = 5$, $C = 0.05$, increasing distribution

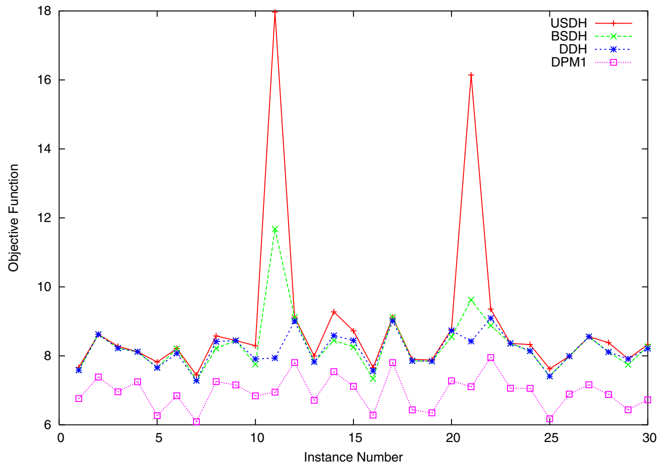

Fig. 7. Objective function value returned by the algorithms, $n = 100$, $p = 5$, $C = 0.05$, triangle distribution

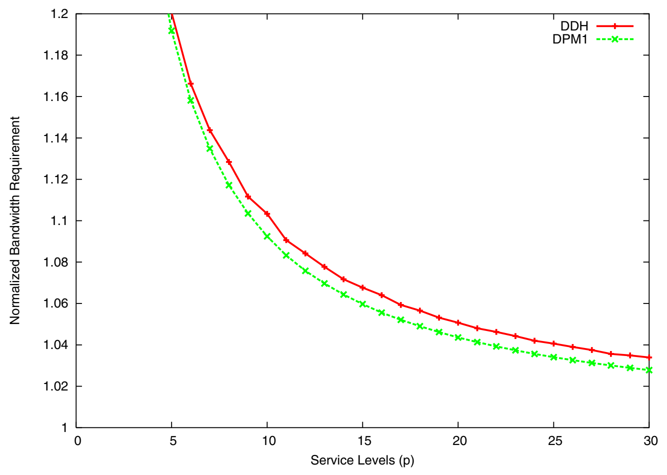

Fig. 8. Normalized bandwidth requirement against p, uniform distribution

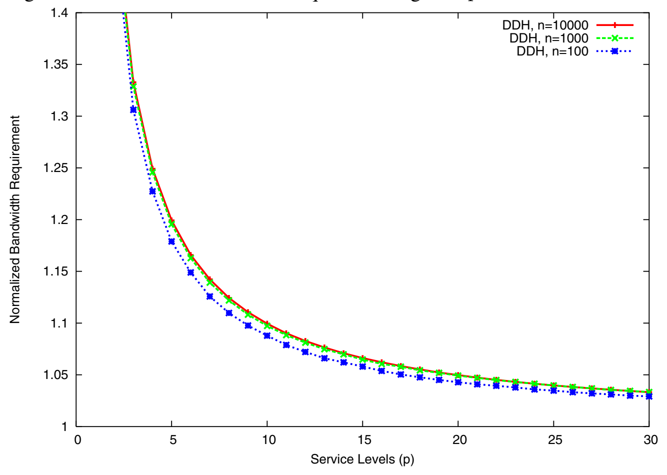

Fig. 9. Normalized bandwidth requirement against p, uniform distribution

similar behavior for all distributions. Also, the normalized bandwidth requirement decreases rapidly with the number of service levels; this result can be explained by noting that as the number of levels becomes very large, the tiered-service network reduces to a continuous-rate network.

Figure 9 shows the effect of the number n of demands on the normalized bandwidth requirement for the DDH algorithm; the effect on the DPM1 algorithm is similar. We observe that as the number n of demands increases, the normalized bandwidth requirement does increase slightly, but the effect diminishes quickly; in fact, the curve for $n = 10,000$ almost coincides with the curve for $n = 1,000$ in the figure. The conclusions we can draw from these figures, and similar ones which can be found in [3], is that (1) with $p = 10 - 15$ levels, the bandwidth required by a tiered-service network is only about 5-10% higher than that of a continuous-rate network; (2) the additional constraint that all service levels be a multiple of a basic unit only slightly adds to the bandwidth penalty; and (3) increasing the number n of demands imposes only an incremental penalty on bandwidth.

Impact on network performance. Finally, let us examine the practical impact of tiered service on overall network performance. To this end, we consider a scenario in which LSPs arrive and depart dynamically. An LSP between source-destination pair (s, d) requires a certain amount of bandwidth; if a path between s and d with sufficient resources can be found, the LSP is established, otherwise, it is rejected (blocked). The performance measure of interest in this context is the LSP blocking probability. We use simulation to compare the blocking probability of a continuous network to that of a tiered-service network. In a continuous network, an LSP requiring bandwidth x_i is accepted if a path with at least that much bandwidth can be found. In a tiered-service network, the bandwidth demand x_i is first mapped to the next highest service level offered, say, z_j, and the LSP is accepted if a path with bandwidth at least equal to z_j is found. The service levels for the tiered-service network are computed in advance for the given demand distribution, using the appropriate algorithm (DPM1 or a CDPM1 heuristic).

In our simulation model, LSP requests arrive as a Poisson process with rate λ, and the mean LSP holding time is an exponentially distributed random variable with rate $\mu = 1$. Each simulation run lasts until 100,000 LSP requests have been served. Each point in the blocking probability curves shown here is the average of thirty simulation runs; we also plot 95% confidence intervals which we estimated using the method of batch means. All the results are for the NSFNet network topology, which can be found in [3]. The capacity of all links is set to two units of bandwidth; since the demand distributions in Table IV are defined in the interval $[0, 1]$, this assumption implies that the bandwidth requested by any LSP is at most one half the link capacity.

Figure 10 plots the blocking probability against the LSP arrival rate for a continuous network and two tiered-service networks, one using DPM1 to obtain the service levels and one using DDH, a representative algorithm for the CDPM1 problem. As expected, the blocking probability of the continuous-rate network is lowest, that of the tiered-service network allocating bandwidth in multiples of a basic unit is highest (DDH algorithm), and that of a network (DPM1 algorithm) which minimizes the excess bandwidth is in between the other two. The higher blocking probability experienced by a tiered-service network is a direct result of the additional resources that such a network uses for each traffic demand. However, the increase in blocking probability is rather small and it may be more than compensated by the advantages of tiered service.

Figure 11 shows the behavior of the blocking probability for the DDH algorithm as we vary the number of service levels p. The curves confirm the intuition that as p increases, the blocking probability of the tiered-service network decreases and tends towards that of a continuous-rate network. This figure suggests that the network designer/engineer may select the number p of the service levels to be offered so as to combine the advantages of tiered service with the performance of a continuous-rate one.

V. Concluding Remarks

Tiered service has many potential applications in networking, especially in contexts where catering to very large sets of heterogeneous users/demands poses significant scalability problems. We have developed a theoretical framework for reasoning about service level selection. Our ongoing research aims to extend this work by (1) quantifying the benefits of tiered service, and (2) including pricing considerations in service level selection.

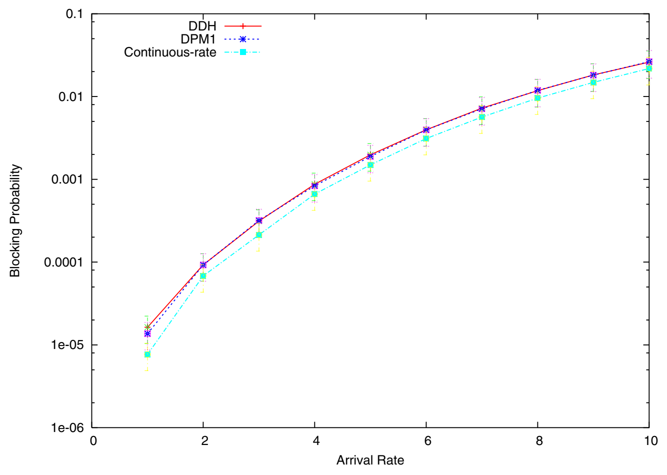

Fig. 10. Blocking probability against the arrival rate, $n = 100,000$, $p = 30$, $C = 0.05$, uniform distribution

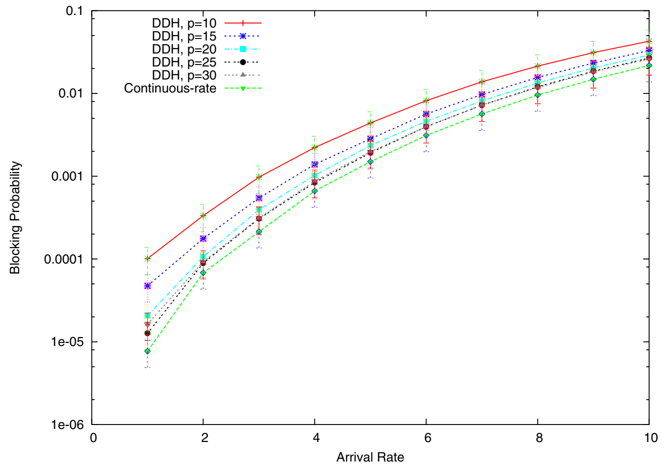

Fig. 11. Blocking probability against the arrival rate, $n = 100,000$, $C = 0.05$, uniform distribution

References

[1] A. Aggarwal, M. Klawe, S. Moran, P. Shor, R. Wilber. Geometric applications of a matrix searching algorithm. *Algorithmica*, 2:195-208, 1987.
[2] A. Aggarwal and J. Park. Notes on searching in multidimensional monotone arrays. In *IEEE FOCS*, pages 497–512, 1988.
[3] Nikhil Baradwaj. Traffic quantization and its application to QoS routing. Master's thesis, NCSU, Raleigh, NC, August 2005.
[4] D. Cavendish, K. Murakarni, S-H. Yun, O. Matsuda, and M. Nishihara. New transport services for next-generation SONET/SDH systems. *IEEE Communications Magazine*, 40(5):80–87, May 2002.
[5] M. Daskin. *Network and Discrete Location: Models, Algorithms, and Applications*. John Wiley and Sons, New York, 1995.
[6] R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operations Res. Letters*, 10:395–402, 1991.
[7] C-T. Lea, A. Alyatama. Bandwidth quantization and states reduction in the broadband ISDN. *IEEE/ACM Tran. Network.*, 3:352-360, June 1995.
[8] N. Megiddo and K. J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Computing*, 13:182-196, Feb. 1984.
[9] S. Ramabhadran and J. Pasquale. Stratified round robin: A low complexity packet scheduler with bandwidth fairness and bounded delay. In *Proceedings of ACM SIGCOMM '03*, pages 239–249, August 2003.