Prioritized Resource Allocation for Stressed Networks

Cory C. Beard, *Member, IEEE,* and Victor S. Frost, *Fellow, IEEE*

***Abstract*—Overloads that occur during times of network stress result in blocked access to all users, independent of importance. These overloads can occur because of degraded resource availability or abnormally high demand. Public broadband networks must dynamically recognize some multimedia connections as having greater importance than others and allocate resources accordingly. A new approach to connection admission control is proposed that uses an upper limit policy to optimize the admission of connections based on the weighted sum of blocking across traffic classes. This results in a simple algorithm suitable for multimedia and packet networks. This work is also the first to demonstrate that the use of an upper limit policy is superior to traditional approaches of adding extra capacity or partitioning capacity, both in terms of the amount of resources required and sensitivity to load variations. An upper limit policy can also be deployed much faster when a large overload occurs from a disaster event.**

***Index Terms*—Computer network performance, resource management.**

I. Introduction

THE PUBLIC data network provides a resource that could profoundly impact high-priority activities to society like defense and disaster recovery operations [1]. Under stress, however, the public network has historically been a virtually unusable resource [2]. Today's public network resource allocation mechanisms do not prioritize the way they allocate resources, instead working on a first-come-first-served basis. Loads on public networks reach up to five times normal during an emergency [3], and important traffic receives equally poor access to resources as low-priority traffic. In a report by the National Research Council [4], this problem was referred to by emergency management experts as the need to give "emergency lane" access to resources.

The purpose of this work is to support ongoing activities in the ITU and IETF on developing the International Emergency Preparedness Scheme (IEPS) [5], [6]. The work here supports IEPS's initial focus on IP telephony, as well as applies to all types of emergency-related multimedia traffic.

Manuscript received December 13, 1999; revised August 10, 2000; recommended by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Liebeherr. This work was supported by the Madison and Lila Self Graduate Fellowship at the University of Kansas. An earlier version of this paper was presented at IEEE MILCOM'99.

C. C. Beard is with the School of Interdisciplinary Computing and Engineering, Department of Electrical and Computer Engineering, University of Missouri-Columbia, Kansas City, MO 64110-2499 USA (e-mail: beardc@umkc.edu).

V. S. Frost is with the Department of Electrical Engineering and Computer Science and the Information and Telecommunication Technology Center, University of Kansas, Lawrence, KS 66044-7541 USA (e-mail: frost@eecs.ukans.edu).

Publisher Item Identifier S 1063-6692(01)08994-4.

Even though development of mechanisms to support different levels of quality of service (QoS) promises predictable service to multimedia connections once they are established, the question of how and to whom to give access to those resources in crisis events has not been fully addressed. Blocking must occur, since it is always possible for resource demand to exceed availability, but the question is how to control that blocking, especially for those who need network resources the most.

A. Problem Context

This work proposes the use of prioritized resource allocation. To accomplish this, an architecture is necessary that consists of two components. The first component recognizes which connections are more important and classifies them with other connections of similar importance. This was addressed in [7], [8] where an architecture of geographically distributed ticket servers was proposed to issue tickets to important connections for use when seeking connection admission. In [9], the architecture was shown through simulation and performance analysis to be implementable and to not introduce prohibitively long connection setup delays.

The second component of the architecture performs the actual resource allocation according to the priority and multimedia demands of the connections. Of particular interest are times of network stress when significant numbers of connection requests must be denied (i.e., blocked) to preserve QoS for other connections and protect the potential for subsequent, more important requests to be admitted.

The approach proposed here is to use an *upper limit* (UL) policy for connection admission that uses resources currently available and sets upper limits on the amount of resources that can be used for each prioritized class. In effect, this approach limits the full use of the capacity, at least by lower priority classes, so that connection requests from higher priority classes are likely to arrive with free resources available. Alternatives to this approach would be to partition resources, to add new capacity when needed, or to allow all connections to be admitted, but then preempt connections from lower priority classes for higher priority requests.

Later sections compare the partitioning and new capacity approaches to the upper limit policy. For typical scenarios, partitioned capacity can cause 50% more blocking compared to a UL approach. Excess capacity approaches require more than double the capacity of a UL approach. New capacity may also take several more hours to implement in the aftermath of a disaster. The benefits of a preemption approach are being studied by the authors but are not discussed here. It should also be noted that in

the U.S., preemption for emergency management activities is not used [6].

The next subsection defines the scope of the problem. Following that is a discussion of related work on connection admission policies and approximations of blocking for those policies.

B. Problem Statement

The basic context for the problem lies in the application of stressed network conditions to the problem of loss networks. In a loss network, requests for connections are either accepted or blocked; no queueing of requests occurs. The network is assumed to use connection-oriented resource allocation to provide levels of QoS, and applies to work being done many areas (MPLS, ATM, TCP/IP/RSVP, etc.). This does not assume, however, that connection-oriented mechanisms are used throughout the network, since scalability concerns might necessitate aggregation or connectionless approaches in backbone networks. It is only assumed that connection-oriented mechanisms are used in access networks to limit the number of connections that inject traffic into backbone networks. This work, however, could still be useful in backbone networks to estimate capacities needed for different classes of traffic, even if connection state were not maintained at every node.

An arbitrary number of traffic classes is allowed here, with each class defined by an importance level and the amount of resources used by each connection. These multimedia connection requests are assumed to arrive according to independent and identically distributed Markov processes. Service times, however, are generally distributed [10]. Estimates for the current overall load and load per class are provided by the ticket server architecture discussed above [8] that tracks resource utilization through the frequency of ticket requests granted. It is assumed that networks can be considered stationary for the periods of time within which load estimates are conducted. This analysis first looks at the case of one communication link and one resource (effective bandwidth), and then is extended for a network of links.

Addressed here is the problem of minimizing the weighted sum of blocking when allocating resources in a network, where

$$W_B = \text{weighted sum of blocking} = \sum_{r=1}^{R} w_r B_r \qquad (1)$$

and

$$\begin{aligned} R &= \text{number of classes} \\ w_r &= \text{weight for class } r \\ B_r &= \text{probability of blocking for class } r. \end{aligned}$$

Classes of traffic are assigned weights that are consistent with the importance of their activities when using the network. During normal operations, those with greater importance may be those which generate greater revenue, but in times of crisis those that deal with emergencies or natural disasters will be more important.

While others have proposed optimization based on maximizing revenue or utilization [11], [12], this work uses a weighted blocking metric to directly control, monitor, and bound blocking probabilities. Controlling blocking gives network operators a more direct understanding than other optimization metrics of the level of service given to specific priority traffic classes. A set of weights reflects the relative cost of blocking for each class. A policy can then be formulated from these weights to minimize W_B. This weighted blocking criteria is used as a basis of comparison between resource allocation approaches.

For upper limit policies to become well accepted for implementation, the following two questions must be answered.

1) Is dynamic prioritization of resources really beneficial? A dynamic prioritization approach requires an architecture to dynamically determine which connections are more important given the current state of the environment (e.g., during disasters). With such an implementation cost, dynamic prioritization must use significantly fewer resources or provide significantly lower blocking.
2) If resources are dynamically prioritized, how would connection admission control (CAC) functions decide which connections to admit? Could algorithms be simple enough for use on standard network hardware?

The approach to addressing these questions was to first answer the second question by developing a simple efficient CAC process for an arbitrarily large number of traffic classes. Such an approach would be used in policy frameworks for network QoS provisioning [13]. Then the first question was addressed by showing the CAC algorithm to provide better utilization of resources and less sensitivity to load variations than traditional approaches.

II. Related Work

In the most general case of resource allocation, all connections are admitted simply if resources are available at the time a connection is requested. This is commonly called a *complete sharing* (CS) admission policy where the only constraint on the system is the overall system capacity, C. In a CS policy, connections that request fewer resource units are more likely to be admitted (e.g., a voice connection will more likely be admitted compared to a video connection). A CS policy does not consider the importance of a connection when resources are allocated.

A. Types of Policies

Other policies have been derived to provide a more equitable balance between users or to provide optimized access to resources. Ross [14] provides extensive discussion of different approaches that have been taken. All policies take the state space (allowable combinations of numbers of connections from each class) from CS and constrain it in some way. Some have derived optimal policies [11], [15]–[19]. To implement optimal policies, however, a detailed accounting may need to be made of every allowable network state and state transition, which is impractical for networks of even modest size. Therefore, a set of generally nonoptimal heuristic policies have been developed that are simpler to implement and provide a more intuitive understanding of how resources are managed. In a *complete partitioning* (CP) policy, every class of traffic is allocated a set of resources that can only be used by that class. A *trunk reservation* (TR) policy

says that class i may use resources in a network up until the point that only r_i units remain unused [12]. A *guaranteed minimum* (GM) policy [20], [21] gives each class their own small partition of resources. Once used up, classes can then attempt to use resources from a shared pool that all classes use. And finally, an *upper limit* (UL) policy [20] places upper limits on the numbers of connections possible from each class to ensure that no one class can dominate the use of resources.

Several comparisons have been made between heuristic policies and with the optimal policy. The upper limit policy was found to be optimal for maximizing revenue over coordinate convex policies [18] (i.e., policies where the product form of Erlang's equation is preserved) of two classes [11] and maximizing revenue over coordinate convex policies of an arbitrary number of classes for asymptotically large links [12]. The CP, GM, UL, and TR policies were found to outperform the CS policy (with respect to maximizing revenue when bounds are placed on blocking for each class) when significant differences between classes existed in requirements for bandwidth and offered load [22]. UL and GM policies were also shown to significantly outperform TR policies, when controlling blocking performance in the presence of temporary overloads that occur before system control parameters can be adjusted [21].

The above policies are effective when network traffic behaves consistent with the loading assumptions made to implement the policies. Recent work, however, has sought to develop policies that are robust when class loading increases beyond engineered loading. Virtual partitioning (VP) [23] uses a variant of trunk reservation, where classes are assigned one trunk reservation level normally (i.e., r_i) but have a different, more stringent one imposed on them when they exceed the nominal capacity allocated to them. These reservation parameters are assigned based on optimizing revenue as a combination of rewards and penalties. The objective is to prevent heavily loaded classes from degrading the performance of those that have loads within their prescribed bounds. In essence, to the benefit of underloaded classes overloaded classes that already experience high blocking from being overloaded are penalized further by having more restrictive trunk reservation imposed. In addition to VP, other work has sought to provide robustness to load variations by explicitly and dynamically controlling buffer occupancy thresholds [24]–[26].

The work here proceeds with further development of the upper limit policy. Fig. 1 illustrates an upper limit policy for two classes of traffic. It shows a linear bound on the number of connections that the CS policy imposes, and examples of additional thresholds for each class imposed by the upper limit policy. A valid upper limit policy need not implement a threshold for every class.

At engineered loads, a UL policy is competitive with TR and optimal policies when blocking performance is to be controlled. When loads deviate from their engineered values, a UL policy accomplishes the same goals and is simpler mathematically and simpler to implement than VP. It imposes upper limits on lower priority classes so their overloads do not affect higher priority classes. For higher priority classes, no upper limits need be imposed at all; if they exceed their engineered loads, they can use whatever additional capacity might be available at the time. As seen in Section IV, the impact of overloaded low-priority classes is then minimal. The UL approach here also is distinct in that it optimizes based on weighted blocking, not revenue.

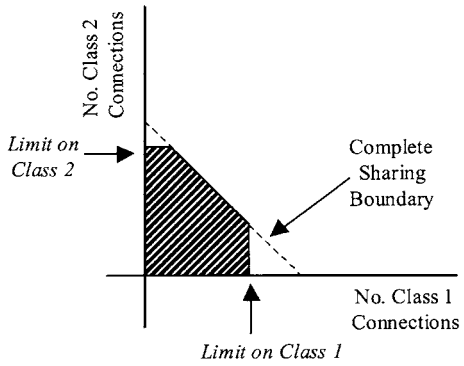

Fig. 1. Illustration of the upper limit policy.

The simplicity of the definition of the UL policy provided opportunities for developing simple optimization algorithms that are presented in the next sections. Later sections show that capacity utilization is not significantly sacrificed and that robustness objectives are met by using the UL policy.

B. Computational Methods

Our work starts from the assumption that effective bandwidths can be assigned to connections in each class that encapsulate the rate, delay, and loss requirements of a flow [23], [27]–[29]. We adopt this approach because it allows blocking analysis to be performed by considering the set of possible network states where all flows can have their desired QoS supported. By using effective bandwidths, the boundary of this state space can be defined by a single linear equation. This approach has been justified to be useful in many contexts [27], even when combining buffer and bandwidth allocation to meet delay, loss, and bandwidth requirements of flows [28].

This work can also serve as a basis for implementing upper limit policies where more accurate boundary characterizations can be developed. Nonlinear equations and systems equations can be derived which provide higher utilization of capacity and tighter control of bandwidth and buffers to meet QoS requirements [28]. This is discussed in Section III-D. For example, see [30], where multiple equations were defined that ensure bounded delay for all connections at a node that uses rate based or deadline ordered schedulers.

Most of the work on computing blocking for CAC policies has centered on the Erlang loss function, which provides the ability to exactly compute blocking for different policies under Markov connection arrival assumptions [12], [15]–[19], but can only reasonably be used when networks are of modest size (less than 1000 units of capacity). In detailed work on upper limit policies, [20], [21] provide a numerical inversion method using generating functions to find exact blocking probabilities for UL and GM policies. They conclude, however, that "the numerical inversion algorithm can also have high complexity … the current upper limit on the dimension (number of classes) amenable for computation is about five" [20]. Optimal UL threshold parameters have been found in [21] and [22], but

only using heuristic search algorithms. The goal here, however, is to find the optimal parameters using direct optimization with no practical limit on the number of classes or the size of the network.

Approximations for the Erlang loss function as networks asymptotically grow in size are particularly useful in this regard. An important result was produced by Kelly [31] and then later expanded by Hunt and Kelly [32]. Kelly approximated blocking probabilities from the expected value of the number of connections in progress in a network. From this result, a class of policies can be defined that all have the same blocking probability. Kelly's theory was used as the basis for [33]–[36], and also the work here. No work had been done to date on policy optimization in the overloaded conditions that would occur after a disaster event, where load and capacity tend to infinity at a constant ratio, while load is greater than capacity.

Thus the first contribution of this research is to find a class of policies that optimize the weighted sum of blocking in overloaded conditions. This is provided in Section III. Then in Section IV, the second contribution is the selection of one of the policies within this class, the upper limit policy, to implement that optimal solution for infinite capacity networks to a practical network. The final contribution, given in Sections V and VI, is the demonstration of the usefulness of the UL policy in practical situations as compared to commonly used alternatives.

III. Asymptotically Optimal Weighted Blocking

This section provides a new derivation for optimal resource allocation in asymptotically large networks based on a weighted blocking objective function. In asymptotically large networks, load and capacity asymptotically approach infinity proportionally at a constant ratio of load to capacity greater than 1 (i.e., an overloaded condition). At first, the system under consideration has a single resource (effective bandwidth), an arbitrary number of classes, and a single link. At the end of the section an extension is provided for an arbitrary number of links.

A. Blocking Probabilities in Asymptotically Large Networks

Kelly's formulation for asymptotically large networks [31] is based on a network where each class of traffic uses an integer number of resources along each link in a path. The analysis assumes that connection holding periods are generally distributed with unit mean [31]. Each class of traffic is defined by the route each connection takes and the amount of resources it uses on each link. A class of traffic is limited in the number of simultaneous connections it can have by complete sharing policies on each link the class traverses, with capacity constraint, C_j, for link j. The constraints for all links on the network are

$$\mathbf{An} \leq \mathbf{C} \tag{2}$$

where row j of $\mathbf{A}$ defines the CS constraint for link j. The vector $\mathbf{n}$ is the number of connections in progress per class, and $\mathbf{C}$ is the vector of link capacities.

Kelly [31] then proceeds to find the most likely state (MLS), $\overline{\mathbf{n}}$, and shows that the normalized expected value of the number of connections in the system asymptotically converges to the MLS [31]. Given λ_r as the average arrival rate per class, the most likely state can be found as the solution to a constrained nonlinear optimization problem. Using Lagrange multiplier methods and converting into a dual problem, the blocking per class can be found from finding the set of y_j's that optimize

$$\min \sum_{r=1}^{R} \lambda_r e^{-\sum_{j=1}^{J} y_j A_{jr}} + \sum_{j=1}^{J} C_j y_j \quad \text{subject to } y_j \geq 0. \tag{3}$$

The variables y_j are the Lagrange multipliers. The MLS (using real numbers) is $\overline{\mathbf{x}}$, and the coordinate for class r of $\overline{\mathbf{x}}$ is

$$\overline{x}_r = \lambda_r \prod_{j=1}^{J} e^{-y_j A_{jr}}. \tag{4}$$

Blocking is found using Little's Law from the MLS to be

$$B_r = 1 - \prod_{j=1}^{J} e^{-y_j A_{jr}} = 1 - \frac{\overline{x}_r}{\lambda_r}. \tag{5}$$

B. Asymptotically Equivalent Policies

Now we use this result to formulate optimization of the weighted sum of blocking. The goal is to find a policy where the most likely state within that policy's state space optimizes the weighted sum of blocking formulation given in (1). Since multiple state spaces can be defined which all have the same MLS, the solution to such a problem will result in a class of policies which optimize weighted blocking. When all policies have the same MLS, the same blocking probabilities and the same weighted sum of blocking will result; hence, the policies can be considered asymptotically equivalent.

Under what conditions will policies have the same most likely states? First of all, if a policy has a most likely state for a state space Ω, another policy will have the same most likely state if the following two conditions are met.

1) The state space of the second policy is a subset of the first policy.
2) The most likely state for the first policy lies within the state space of the second policy.

These conclusions come from an understanding of the Erlang loss function which is defined as

$$\pi(\mathbf{n}) = G \prod_{r=1}^{R} \frac{\lambda_r^{n_r}}{n_r!}, \qquad \mathbf{n} \in \Omega$$
$$G = \left(\sum_{\mathbf{n} \in \Omega} \prod_{r=1}^{R} \frac{\lambda_r^{n_r}}{n_r!} \right)^{-1} \tag{6}$$

where here it is assumed that mean holding times are equal to one. The probability of being in a certain state is $\pi(\mathbf{n})$, when $\mathbf{n}$ is within the state space Ω for a given policy.

The most likely state, $\overline{\mathbf{n}}$, comes from finding the maximum value of $\pi(\mathbf{n})$. If a new state space Ω' is formed as a subset of the original Ω, the values of $\pi(\mathbf{n})$ in Ω' change uniformly in the multiplier, G. If $\overline{\mathbf{n}} \in \Omega'$, then $\overline{\mathbf{n}}$ is the most likely state of Ω'. No new states have been added and a change in the multiplier G does not affect the location of the most likely state.

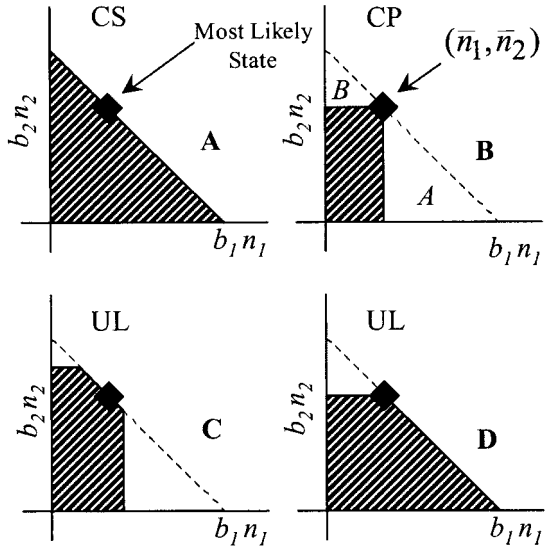

Fig. 2. Illustration of asymptotically equivalent CAC policies.

TABLE I
BLOCKING PROBABILITIES FOR ASYMPTOTICALLY EQUIVALENT POLICIES

C=1000 b_1=1 λ_1=181 b_2=2 λ_2=909

Policy	L_1	L_2	B_1	B_2
Approx			0.3065	0.5190
A, Actual	∞	∞	0.3071	0.5202
B, Actual	125	875	0.3206	0.5203
C, Actual	200	900	0.3065	0.5202
D, Actual	∞	875	0.2688	0.5242

The base state space from which to start (i.e., Ω from above), is the CS policy. This provides the largest possible state space for a given capacity. In overloaded conditions, the most likely state will lie on the boundary imposed by the overall capacity. For policies which are subsets of Ω, but which include the CS MLS, the blocking will be the same. Fig. 2 shows examples of policies that share the same MLS with the CS policy. Example coordinates of the MLS are $\overline{n}_1 = 125$ and $\overline{n}_2 = 437$, where values of $\overline{n}_r$ are integers rounded from the values of $\overline{x}_r$ found from (4). Table I shows in the first row the computed blocking probabilities from the asymptotic approximation that will apply to all policies. These are compared to the actual blocking probabilities computed using Erlang's loss function for each policy in Fig. 2. The values of L_r in the table are upper limits imposed on each class in addition to the overall capacity constraint, such that

$$b_r n_r \leq L_r.$$

To change blocking probabilities, we require policies which are a subset of original Ω, but which do not include the CS MLS. Note that all possible policies will be a restriction on the CS state space, but later sections demonstrate that the lowered utilization of capacity is not significant for the cases considered here.

C. Asymptotically Equivalent CS and CP Policies

To find policies which optimize weighted blocking asymptotically, we start with a CS policy, show that a CP policy can be created that is equivalent to it, and change the CP policy to improve weighted blocking over the CS policy. Then we show that the optimal CP policy is optimal over all policies and define the class of policies that are equivalent to the optimal CP policy. This discussion is provided in the following subsections.

Consider a CP policy [Fig. 2(b)]. A CP policy could be formulated to be asymptotically equivalent to a CS policy that had its MLS on the state space boundary, as long as the "corner" of the CP region corresponded to the CS MLS. In overloaded conditions the MLS for the CP policy will lie on the corner of CP state space. Weighted blocking, therefore, can be optimized by modifying the location of the CP corner.

First, consider some properties of the asymptotic CP policy. To use Kelly's results from [31] for a CP policy on a single link, the form of the constraints, $\mathbf{An} \leq \mathbf{C}$, must be changed. Instead of defining $\mathbf{A}$ by CS constraints per link, $\mathbf{A}$ is formulated as a set of constraints that restrict the usage of each class to the amount of capacity in its partition, where

$$b_r n_r \leq C_r \tag{7}$$

C_r is the capacity in each partition, and

$$\sum_{r=1}^{R} C_r = C. \tag{8}$$

The form of the matrices then becomes

$$\mathbf{An} \leq \mathbf{C}$$

$$\begin{bmatrix} b_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & b_2 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & b_{R-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & b_R \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ \vdots \\ n_R \end{bmatrix} \leq \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_{R-1} \\ C_R \end{bmatrix}. \tag{9}$$

The result is Kelly's same optimization problem over a state space that is constrained by $\mathbf{An} \leq \mathbf{C}$, now with $\mathbf{A}$ defined by thresholds from a complete partitioning policy.

Using (3) results in blocking for class r of

$$B_r = 1 - \frac{C_r}{b_r \lambda_r} \tag{10}$$

subject to

$$0 \leq C_r \leq b_r \lambda_r \quad \text{and} \quad \sum_{r=1}^{R} C_r = C. \tag{11}$$

Details of the derivation are given in [7].

This result is surprisingly simple and very useful. Since λ_r and b_r are constants, the blocking probability of a class r connection is a simple linear function of size of the partition for that class, C_r. The interaction between blocking for different classes is through

$$\sum_{r=1}^{R} C_r = C$$

which indicates that for every increase in C_r to lower blocking for one class, one or more other classes must decrease their C_r

and experience an increase in blocking. As C_r increases for a particular class, its blocking can go to zero.

While it is reasonable to allow blocking for one class to increase to provide better blocking for another class, it is also useful to set limits on blocking for any particular class. The constraints on C_r from (11) can be modified using (10) with the following constraints on blocking probabilities:

$$B_{r,\min} \leq B_r \leq B_{r,\max} \tag{12}$$

to result in the constraints on C_r

$$b_r\lambda_r(1 - B_{r,\max}) \leq C_r \leq b_r\lambda_r(1 - B_{r,\min}). \tag{13}$$

In this work, we are concerned with overloaded conditions. It is shown in Appendix A that in such conditions, the MLS for the CS state space always lies on the CS boundary. It is also shown in Appendix A that the MLS for a CP policy will also be on the CS boundary (i.e., on the "corner" of the CP region), if

$$b_r\lambda_r > C_r \tag{14}$$

for all r. This means that in addition to all classes together overloading the capacity, each class overloads its partition. Note that in (13), this already serves as a constraint to the asymptotic blocking approximation; therefore, the approximation can be considered as applying to overloaded conditions.

D. Asymptotically Optimal CP Weighted Blocking

With (10), (11), and (13), an optimal CP policy can be derived to minimize the weighted sum of blocking. This can be viewed as starting from a CP policy that is equivalent to the CS policy (from the previous subsection) and modifying the CP policy to improve weighted blocking. Starting with the weighted blocking metric in (1), the following linear program is formed to optimize partitions

$$\begin{aligned} \min \quad & \left(-\sum_{r=1}^{R} a_r C_r\right) \\ \text{subject to} \quad & C_r + s_{r,1} = C_{r,\max} = b_r\lambda_r(1 - B_{r,\min}) \\ & C_r - s_{r,2} = C_{r,\min} = b_r\lambda_r(1 - B_{r,\max}) \\ & \sum_{r=1}^{R} C_r = C \\ & C_r,\ s_{r,1},\ s_{r,2} \geq 0. \end{aligned} \tag{15}$$

Variables $s_{r,1}$ and $s_{r,2}$ are slack variables for the inequality constraints in (13), and a_r is the ratio of weight to load for class r, found from

$$\begin{aligned} v_r &= b_r\lambda_r = \text{load for class } r \\ a_r &= \frac{w_r}{b_r\lambda_r} = \frac{w_r}{v_r} = \text{load adjusted weight for class } r. \end{aligned} \tag{16}$$

In essence, a_r is a new load-adjusted weight used for the optimization in (15) which is the original weight divided by the class load. This concept of load-adjusted weights is discussed more in Section IV-B where guidelines for selecting weights for priority classes are discussed.

Note that (15) includes the constraint for partition sizes from (8)

$$\sum_{r=1}^{R} C_r = C$$

which is derived from the fact that the partition for each class is defined as

$$b_r n_{r,\max} = C_r$$

and the boundary of the state space is defined by a single equation that defines the overall capacity of the system in terms of effective bandwidths as

$$\sum_{r=1}^{R} b_r n_r \leq C.$$

If a more detailed model of system bandwidth and buffer allocation were used, the constraints for the partitions might include a system of equations, some of which might be nonlinear. In such cases, for example in [30], the form from (15) would be the same, except for changes to the form of the constraints for partition sizes.

For the case here, all that remains is to show that the CP policy that results from the above linear program optimizes weighted blocking over all policies, not just over all CP policies. The MLS for the CP state space must be that state which optimizes weighted blocking in the CS state space from which CP is a subset.

Consider the optimal CP policy. The MLS is located at $\overline{n} = (\overline{n}_1, \overline{n}_2, \ldots, \overline{n}_R)$, and the weighted blocking, W_B, is computed from (1) and (5) as

$$W_B = \sum_{r=1}^{R} w_r \left(1 - \frac{\overline{n}_r}{\lambda_r}\right). \tag{17}$$

To reduce the minimum W_B, one of the adjacent states to $\overline{n}$ must produce lower weighted blocking. Since $\overline{n}$ is on the CS boundary, all states adjacent to $\overline{n}$ but beyond the CS boundary cannot be considered. If a state is chosen where numbers in no class increase, then (17) will not decrease. If some numbers increase and others decrease, then these states will be on or close to the CS boundary. For those states near but not on the CS boundary, there will always exist another state that lies on the CS boundary that will produce lower W_B than that state, because more connections could still be supported. Therefore, only states that lie on the CS boundary are candidates for improving W_B. All such states, however, would have been considered in the optimization in (15). The state $\overline{n}$ would have already been chosen to produce the lowest weighted blocking. Therefore, the result from (15) is the asymptotically optimal policy for minimizing weighted blocking, not just over the set of possible CP policies, but over all policies.

E. Weighted Blocking Optimization Algorithm

The linear program given in (15) can also be formulated as the following algorithm.

1) Compute all $a_r = w_r/v_r$ and sort in descending order.
2) Allocate the minimum C_r to each class.

3) If $\sum_{r=1}^{R} C_{r,\text{allocated}} > C$, stop. No feasible solution is possible for this set of minimum C_r's. Constraints on the minimum C_r's come from $C_{r,\min} = b_r\lambda_r(1-B_{r,\max})$, so maximum blocking probabilities must be higher or loads $(\lambda_r b_r)$ lower for a feasible solution to exist.
4) Find the remainder of C that can still be allocated,

$$C_{\text{remaining}} = C - \sum_{r=1}^{R} C_{r,\text{allocated}}.$$

5) Find the class, r, which has the largest a_r.
6) Form a new C_r for that class by either allocating all of $C_{\text{remaining}}$ or increasing C_r to its upper limit, whichever would increase C_r the least, according to

$$C_r = C_{r,\text{allocated}} + \min(C_{\text{remaining}}, C_{r,\max} - C_{r,\text{allocated}}).$$

7) Update $C_{\text{remaining}}$.
8) If $C_{\text{remaining}} = 0$, stop. The set of C_r's is the optimal solution.
9) If $C_{\text{remaining}} > 0$, move down the list of a_rs to the next class. If no more classes exist, stop. No feasible solution is possible since the sum of the maximum C_rs is less than C. Constraints on maximum C_rs come from $C_{r,\max} = b_r\lambda_r(1 - B_{r,\min})$, so minimum blocking probabilities must be lower or loads $(\lambda_r b_r)$ higher for a feasible solution to exist.
10) Otherwise, go back to step 5.

The above new algorithm is simple to implement on standard network hardware. Proof that this algorithm produces the optimal solution comes from the fact that no modifications to the set of C_rs produced from the algorithm would produce a better W_B. Those classes where an increase in C_r would improve W_B are already at their maximum, while classes where a decrease in C_r would improve W_B are already at their minimum. Details can be found in [7].

Note also that the effect of this algorithm is to attempt to produce minimum blocking for as many high-priority classes as possible, since (13) shows that selection of $C_r = C_{r,\max}$ would produce $B_{r,\min}$. For one class, blocking would be between maximum and minimum bounds, and all remaining classes would have blocking at their upper bounds. This is consistent with what one would expect from the results of a linear program.

F. A Class of Asymptotically Optimal Policies

Once a complete partitioning policy is found that optimizes weighted blocking, many other asymptotically equivalent policies can be created that would also optimize weighted blocking. These would form a class of policies, where each shared the same most likely state, and, hence, the same weighted blocking.

Fig. 2(b) shows an example of an optimized CP policy for two classes. It also shows the CS policy state space from which it is a subset and illustrates the MLS. Additional asymptotically equivalent policies might be formed by removing states from the shaded region or adding states from regions A or B. Removing states, however, would be illogical since it would further constrain admission. Therefore, it is important to only consider adding states from regions A or B while keeping the same MLS.

In region A, states would have larger n_1 coordinates and smaller n_2 coordinates than the MLS. A state in the region, $\mathbf{n_A}$, would have coordinates of the form

$$\mathbf{n_A} = (n_{1,A}, n_{2,A}) = (\overline{n}_1 + k_1, \overline{n}_2 - k_1 b_1 - k_2) \tag{18}$$

where k_1 and k_2 are integers and the term for $n_{2,A}$ reflects the fact that class 2 must at least give up enough capacity to support k_1 class 1 connections. This new state must be less likely than the MLS, or else it would become the new MLS. Appendix B demonstrates that if the following two relationships are true

$$\frac{\lambda_1 \prod_{i=1}^{b_1} (\overline{n}_2 - b_1 + i)}{\lambda_2^{b_1}(\overline{n}_1 + 1)} < 1 \tag{19}$$

and

$$\overline{n}_2/\lambda_2 < 1 \tag{20}$$

then all of the states in region A have lower likelihood and some or all could be added to the CP state space to create another policy. Using the same approach for region B, if the following two relationships are true

$$\frac{\lambda_2 \prod_{i=1}^{b_2} (\overline{n}_1 - b_2 + i)}{\lambda_1^{b_2}(\overline{n}_2 + 1)} < 1 \tag{21}$$

and

$$\overline{n}_1/\lambda_1 < 1 \tag{22}$$

then all of the states in region B have lower likelihood and some or all could be added to the CP state space to create another policy. Note that if both regions A and B could be added to the CP policy, the result would be a CS policy and a CP policy would have created no improvement.

The approach used here for two classes can be extended to multiple classes using the same method. A later section will discuss which of these asymptotically equivalent policies would be most useful in practical implementation, where capacities are not infinite and the policies are not exactly equivalent.

G. Asymptotically Optimal Policies Over Multiple Links

All of the above development has been based on a single link. This approach can readily be extended to multiple links. In such a case, a traffic class would be defined as before by an equivalent bandwidth requirement b_r and a priority weight w_r. In addition, a class would be defined as having all connections have the same endpoints over a fixed route. Therefore, if a network is composed of J links, each class r connection would use b_r units of capacity on a subset of those links between the source and destination, and zero capacity on the other links.

A complete partitioning policy would then be adopted that limits the number of connections for each class. The partitions would be selected to optimize weighted blocking within the constraints imposed by the capacity on each link. Using a slight change in notation from before, the partition size for class r is

defined as $C_{p,r}$ and the capacity on link j is defined as $C_{l,j}$. If we define a parameter

$$f_{j,r} = \begin{cases} 1 & \text{if class } r \text{ uses link } j \\ 0 & \text{if class } r \text{ does not use link } j \end{cases} \tag{23}$$

then for link j,

$$\sum_{r=1}^{R} f_{j,r} C_{p,r} \leq C_{l,j} \tag{24}$$

or generally in matrix form

$$\mathbf{F}\mathbf{C_p} \leq \mathbf{C_l} \tag{25}$$

where $\mathbf{F}$ is the matrix of $f_{j,r}$ values, and $\mathbf{C_p}$ and $\mathbf{C_l}$ are the vectors of partitions and link capacities. The linear program from (15) to create optimal weighted blocking then becomes

$$\begin{aligned} \min \quad & \left(-\sum_{r=1}^{R} a_r C_r \right) \\ \text{subject to} \quad & C_r + s_{r,1} = C_{r,\max} = b_r \lambda_r (1 - B_{r,\min}) \\ & C_r - s_{r,2} = C_{r,\min} = b_r \lambda_r (1 - B_{r,\max}) \\ & \mathbf{f}\mathbf{C_p} \leq \mathbf{C_l} \\ & C_r,\ s_{r,1},\ s_{r,2} \geq 0. \end{aligned} \tag{26}$$

Note that the form of the linear program is virtually identical to (15), except that partitions must be small enough to be supported by each link on a path. Note also that the partition for a class is the same on every link. Therefore, admission control need only be implemented at the first link (i.e., at an edge node), not on every link, because a connection that can be admitted on the first link will automatically be eligible to be admitted on all links on the path, since a partition is allocated for that class on each link.

Once this linear program has been used to produce an optimal CP policy over multiple links, the same approach as above can be used to find other policies which are asymptotically equivalent.

IV. Practical Systems Using an Upper Limit Policy

The above analysis applies to networks where the load and capacity asymptotically approach infinity. This next section considers realistic systems where the capacity is finite. It addresses the actual implementation of a policy that has been optimized based on a weighted blocking metric. In such cases, capacity is not infinite, so policies that are equivalent asymptotically will not be equivalent. The policies to consider for implementation range from CP where all classes have limits to UL policies where some classes have no limits imposed.

First of all, it is important to consider reasons for not simply using a partitioning approach. Complete partitioning has traditionally been the approach used to provide disaster response communications [37]. This approach is attractive because it effectively creates pools of resources for priority users that are separated from the general public. The public network is usually unable to adequately support defense and disaster recovery communications because it becomes so overloaded that access to resources is virtually impossible for all users. A partitioned set of resources is immune to overloads from the general public.

The problems with such a partitioning approach are twofold, however, as exemplified in the following quote.

> "Radio systems designed and used by emergency management agencies appear to be virtually unused on a day-to-day basis, yet when a major event occurs, these same systems are inadequate for meeting the need to communicate." [32]

Thus the two problems are:

- wasted, unused resources on a day-to-day basis;
- not enough resource access (i.e., high blocking) during major events because of large load increases. CP keeps other classes from using the high-priority partition, but also keeps high-priority traffic confined only to the resources in that partition.

Other policies, for example, UL policies, can increase resource utilization to address the first problem. These are created by including states from side regions A or B from Fig. 2(b) as already discussed. In the most extreme case, an upper limit (UL) policy could be created where all the states in a region A or B are included, effectively creating upper limits for some classes and no upper limits at all for other classes (i.e., the higher priority classes). See Fig. 2(d), for example, where class 1 could be a higher priority class where no upper limit was imposed. An upper limit policy would share as much of the resources as possible and only impose limits on lower priority classes.

The UL policy can address the second problem in two ways. If an existing resource management system using CP cannot dynamically adjust the resources allocated to each class (e.g., because the partitioned resources use separate physical facilities), UL is better simply because it can dynamically adjust. A UL policy would already be using a shared resource, and adaptation to load changes would only involve defining new thresholds for each class. If the CP system can dynamically adjust, however, the UL policy is still superior because a UL policy would not implement thresholds for every class. For those classes without thresholds, the UL policy would be less sensitive to load changes.

Fig. 3 compares CP and UL policies and shows the impact when loading for a high-priority class increases beyond the base loading for which the current thresholds were defined. The behavior under consideration occurs before new load estimates can be used to compute new threshold parameters.

Since a UL policy does not impose a limit on the higher priority class, its blocking does not increase as significantly as with a CP policy. Blocking for the UL and CP policies is compared based on $v_1/v_2 = 0.1$ and $w_1/w_2 = 2$. At base loading, the overall load is 1.5 times the capacity and results in blocking for the high-priority class of about 0.1. The ratio of class 1 load to base class 1 load is then varied from 1 (equal to its base load) to 10 times its base load. Load for class 2 remains fixed. Blocking probabilities start below 0.1 and then grow sharply as load increases. Until the load ratio reaches 1.5, the two policies provide about equal blocking. Once the load ratio increases beyond 1.5, however, blocking for the CP policy is up to 50% higher than for the UL policy.

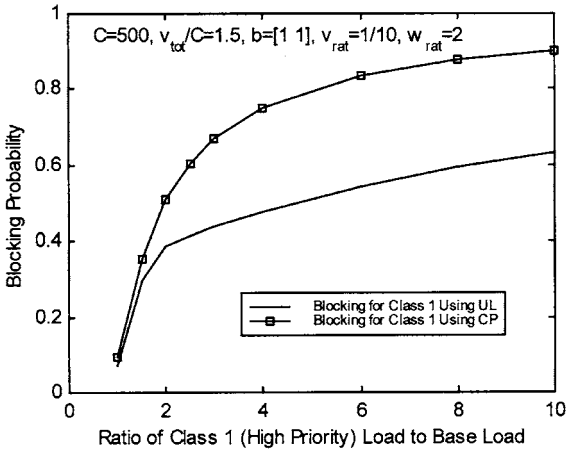

Fig. 3. UL versus CP high-priority blocking for changes to only high-priority load.

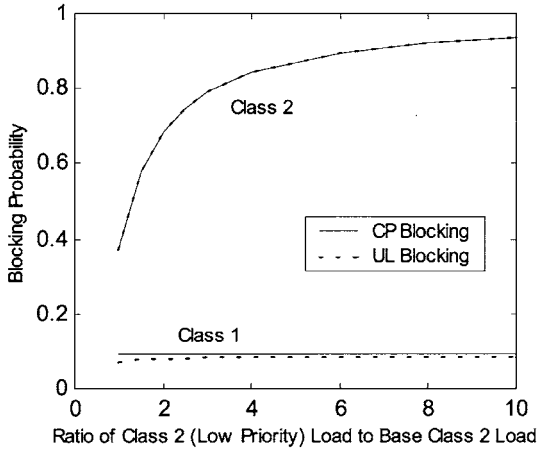

Fig. 4. UL versus CP blocking for changes to only low-priority load.

Fig. 4 shows how blocking changes when only the low-priority load fluctuates. The same loading assumptions as Fig. 3 are used. As load increases, blocking for high-priority class 1 increases somewhat for the UL policy, whereas CP blocking does not change at all because it is using a separate partition. Because the UL policy does share capacity between classes, fluctuations in low-priority load will affect blocking somewhat. Notice, however, that UL blocking starts lower than CP blocking and then approaches CP blocking as low-priority load increases. This demonstrates the fact that UL policies act like CP policies at high loads, and more like sharing policies at low loads. For class 2 blocking, the curves for the CP and UL cases are so close they cannot be distinguished from each other.

This figure also provides insight into the use of a UL policy compared to the virtual partitioning policy discussed in Section II [23]. As low-priority load fluctuates beyond its engineered loading, blocking also increases for the high-priority class. Virtual partitioning uses a trunk reservation approach and seeks to remedy this increase in high-priority blocking by imposing a stricter trunk reservation limit on the low-priority class. This concept of imposing a stricter limit on a misbehaving class could also be implemented using an upper limit policy. This does not appear necessary, however, since Fig. 4 shows

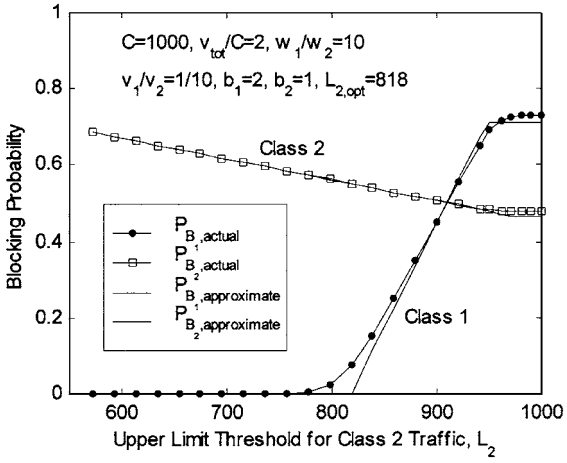

Fig. 5. Blocking variation as the upper limit on class 2 (L_2) changes.

that the UL policy by itself already provides shielding from most of the effects of low-priority load fluctuations.

In summary, load fluctuations will affect CP performance much more than UL performance, so a UL policy is preferable for implementation. A CP policy is only effective if partitions are set to match the exact traffic loading. If a major event causes abrupt load changes, however, the CP policy loses its effectiveness. Also, if a CP policy is implemented from inaccurate load measurements, it becomes less effective. A UL policy overcomes such limitations.

A. Implementation of an Upper Limit Policy

To implement an upper limit policy, the linear program in (15) or the optimization algorithm in Section III-E is first used to find an optimal CP policy. Once optimal partitions are found, the policy can then be converted into an asymptotically equivalent upper limit policy by first setting upper limits L_r on the number of connections per class equal to the partition sizes C_r from the optimization process. Then if class k meets the condition

$$\frac{C_r}{b_r\lambda_r}\left(\frac{C_k}{b_k\lambda_k}\right)^{-b_r/b_k} \leq 1, \qquad \forall r \neq k \tag{27}$$

then the upper limit for that class can be removed. If class k meets the condition $C_k = b_k\lambda_k$, then its upper limit can automatically be removed. See Appendix C for derivation details.

Optimization results for an example UL policy with two classes are provided in Figs. 5 and 6. The link is overloaded at a ratio of overall load (v_{tot}, the sum of v_r) to capacity of 2. The weight of the high-priority load (class 1) is ten times that of class 2 and its load is 1/10 that of class 2. The plots show how the blocking probabilities for each class change as the upper limit on class 2 (L_2), changes. The optimal value is $L_{2,\,opt} = 818$. No upper limit constraint is imposed on class 1.

Fig. 5 shows approximate blocking probabilities compared to the upper limit for class 2; as L_2 decreases from $L_2 = 1000$, blocking for class 2 increases gradually while blocking for class 1 drops sharply. Because the high-priority load is smaller, small changes in L_2 make a bigger impact on blocking for that class. The flat parts of the curves denote the areas where the UL policy is equivalent to a CS policy; changes in UL thresholds do

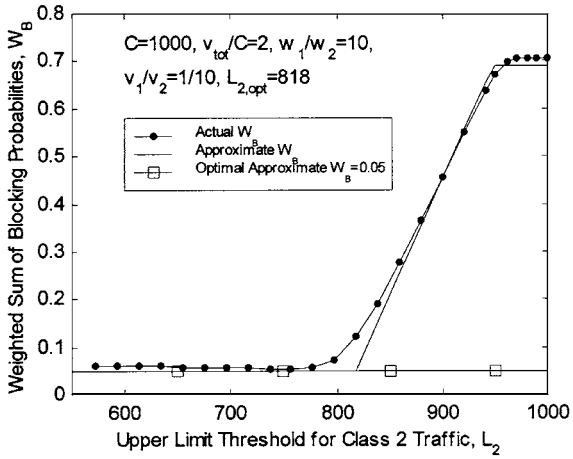

Fig. 6. Weighted blocking variation (W_B) as the upper limit on class 2 (L_2) changes.

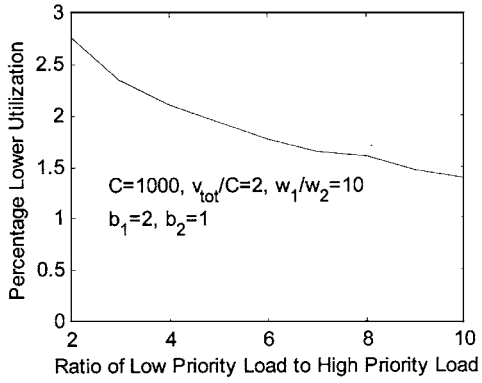

Fig. 7. Percentage reduction in utilization by using an upper limit policy.

not change the most likely state nor the blocking probabilities. The optimization process makes the blocking for class 1 go to approximately zero. Fig. 5 also shows actual blocking using the Erlang loss function. Most notable is the deviation between approximate and actual values for blocking for class 1 in the area of $L_2 = 800$. For all other areas, approximate values are close to actual values. The error in the approximation is on the order of $1/\sqrt{C}$, where C is the overall capacity of the link. In practice, L_2 might be set lower than $L_{2,opt} = 818$, (e.g., $L_2 = 780$), which would make actual blocking for class 1 nearer to zero and blocking for class 2 slightly higher.

Fig. 6 shows how the weighted sum of blocking, W_B, changes with L_2. By decreasing L_2 to decrease the blocking on class 1, W_B drops off sharply, since class 1 is weighted more highly. The gradual increase in blocking for class 2 does not significantly affect W_B.

The significance of these results is seen in the reduction in weighted blocking that occurs. By implementing the UL policy, W_B is reduced from 0.7 to 0.05, only 7% of the weighted blocking as without the UL policy (i.e., a CS policy). This is because blocking for high-priority traffic goes from 0.75 to approximately 0. The upper limit policy caused the blocking for the low-priority traffic to rise from 0.48 to 0.52, a reasonable penalty.

B. Selection of Weights

The use of a weighted blocking optimization function in (1) provides the opportunity for network providers to balance the service provided to different classes of customers by direct knowledge and manipulation of blocking probabilities. Successful weighted blocking optimization, however, is contingent upon effective selection of blocking weights. Note that the main consideration (stated in Section III-E) is that weights ultimately determine which classes receive blocking at their minimum bound, which receive blocking at their maximum bound, and the one class which receives blocking somewhere in between upper and lower bounds.

For classes that are considered the highest priority, a weight with a very large value (even infinity) could be used. Optimization results are not sensitive to the specific selection of the weights for the highest priority classes. The linear program in (15) or the algorithm in Section III-E provides as many of those classes as possible with minimum blocking. In the same way, weights for the lowest priority classes could be set to zero; the algorithm would make as many of those classes as necessary have blocking at their upper bound.

For classes of medium priority, the optimization approach takes into account not only the assigned weights, but also the load for those classes. Classes are ultimately ordered in the algorithm based on their load-adjusted weight, a_r, in (15) and (16). Classes with higher weights, w_r, may be given lower precedence in the algorithm if their loads are high, since their a_r value could be lower than another class even if their w_r value were higher.

Two approaches are suggested to assign weights for medium-priority classes. The first alternative would be to assign relative priorities using w_r *a priori* (for example, using values 1 through 10), and then let the a_r values serve as arbitration mechanisms between classes. Even if some classes might have a higher *a priori* priority, w_r, the a_r values could indicate that it would be more costly to provide those classes with preferred blocking performance, since their loads would be higher. The second alternative would be to assign weights so that the ordering of load-adjusted weights, a_r, would never result in an ordering different than those for w_r. Strict prioritization could then be implemented without respect to class loading. These weights could be assigned on a logarithmic scale, for example, so that one class would have a weight 10 or 100 times that of the next lower priority class. Even if that lower priority class had load much lower, its load-adjusted weight, a_r, would not be higher.

C. Resource Utilization

A valid concern in the use of an upper limit policy (or anything other than a CS policy for that matter) is reduced capacity utilization because of artificial limits imposed on some traffic classes. Setting limits that are well short of the capacity would seem to significantly hamper sharing during normal loading. This is not the case, however.

First of all, it is helpful to consider capacity utilization in overloaded conditions. Fig. 7 shows the reduction in capacity utilization that would be caused by using an optimized upper limit policy in conditions where the overall load is twice the ca-

pacity. In such cases, a CS policy would provide 99.9% utilization, regardless of the balance of loading between classes. Fig. 7 then shows how much less the utilization would be for a UL policy, based on various ratios of low-priority load to high-priority load. As the ratio decreases, lowered utilization increases, but only to 3% in this example (i.e., down to 97% capacity utilization). For the benefits seen above in how W_B can be reduced so significantly, such a small reduction in capacity utilization seems to be a reasonable cost.

During normal operations, network loading would be at less than capacity. During such times, the most important consideration would be the effect of load fluctuations. One might choose to implement a UL policy to provide limits that assume load fluctuations. For example, if a network's normal average loading is 75% of capacity, UL thresholds could be set assuming twice that (i.e., assuming loading at 150% of capacity). When loads were at the base loading levels of 75%, capacity utilization would be at 75% and blocking probabilities would be very low. The upper limits would not cause any less capacity utilization compared to a CS policy. When loads fluctuated even to twice their base level, the UL policy could still ensure that the optimal W_B (or better) was provided.

Consider a specific example where $C = 1000$, $b_1 = 2$, $b_2 = 1$, $v_1/v_2 = 1/10$, $w_1/w_2 = 2$, and $v_1 + v_2 = 750$. Use of a CS policy would result in average capacity utilization of 750.00 and blocking of 3×10^{-16} and 2×10^{-16}. An upper limit policy designed for loading of 1500 (150% of capacity) would set an upper limit on class 2 of $L_2 = 864$ and yield capacity utilization during normal conditions of 750-7.2 $\times 10^{-5}$ and blocking of 2×10^{-18} and 3×10^{-12}. As seen in this simple example, as long as normal loading is not already nearing capacity limits, a UL policy will perform no worse than a CS policy; when load fluctuations occur the UL policy will enforce weighted blocking optimization criteria.

V. Comparison of Excess Capacity and Upper Limit Policies for Practical Systems

An asymptotic approximation that allows optimization of upper limit thresholds has been found. While it has advantages over other resource allocation policies and is efficient to implement, it still must be considered against traditional resource management approaches, e.g., using excess capacity in the network to control blocking, either by overbuilding or by deploying new capacity as the need arises.

A. Numerical Comparisons of CS and UL

The key issue is not whether a CS policy could be implemented to provide the same weighted blocking as a UL policy, but rather *how much* capacity would be required to adhere to this goal. Fig. 8 shows how much CS capacity is needed to provide comparable weighted blocking to a UL policy at various overloads for two classes of traffic. The x-axis denotes the amount of load compared to the base capacity, and the y-axis shows the amount of new CS capacity that would need to be installed with respect to the base capacity. Fig. 8 suggests that a linear relationship exists between the level of overload and the extra CS capacity required. This indicates that for an increase in load, a proportional amount of new CS capacity would have to be installed to keep W_B the same as with a UL policy.

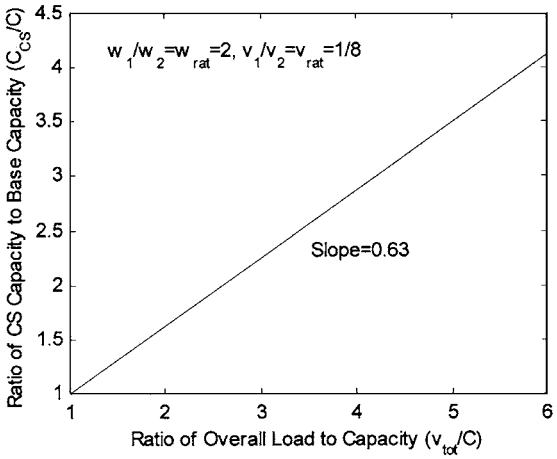

Fig. 8. Extra CS capacity needed versus a UL policy in overloaded conditions.

For the case where two classes of traffic are involved and both classes use the same amount of bandwidth, this linear relationship can be derived analytically using the asymptotic blocking approximation method from [31] in (3). The expression for the ratio of CS capacity to the original capacity, C_{CS}/C, given a level of overload, $v_{\rm tot}/C$, is

$$\frac{C_{CS}}{C} = \frac{v_{\rm tot}}{C}\left(\frac{w_{\rm rat} - v_{\rm rat}}{1 + w_{\rm rat}}\right) + \left(\frac{1 + v_{\rm rat}}{1 + w_{\rm rat}}\right). \tag{28}$$

This is a linear equation in $v_{\rm tot}/C$ where $w_{\rm rat}$ and $v_{\rm rat}$ are constants that signify the ratio between weights and loads for the two classes. The derivation of this equation is in [7].

The slope of the line in Fig. 8, which we call the *increment ratio*, is defined from (28) as

$$I_r = \frac{w_{\rm rat} - v_{\rm rat}}{1 + w_{\rm rat}}. \tag{29}$$

In Fig. 8, the increment ratio is 0.63. A 100% increase in load would require 63% more new capacity; 160% more load would require 100% more capacity.

The equation for the increment ratio in (29) is a linear equation in $v_{\rm rat}$ and a nonlinear equation in $w_{\rm rat}$. The increment ratio changes as the ratio between weights, $w_{\rm rat}$, is varied and I_r has a limit as $w_{\rm rat}$ tends to infinity of $I_r = 1$. This asymptotic limit means that a CS policy will never need to increase capacity more than the amount load has increased.

B. Typical Conditions

The previous subsection provided a comparison of required capacity for CS and UL policies for practical nonasymptotic cases. The remaining issue to consider, however, is what range of overloads, weights, and load ratios could be expected in a crisis scenario.

It is reasonable to assume a network could have loads up to five times its capacity [3] and that high-priority load would never be greater than the total capacity of the network. It is also reasonable to assume that high-priority load will be less than load for low-priority traffic. Even in emergencies where the high-priority load increases dramatically, the demand for low-priority traffic increases as well.

For values of $w_{\rm rat} > 5$ (weight for the high-priority class at least 5 times that for the low-priority class) and $v_{\rm rat} < 1$ (load for the high-priority class less than load for the low-priority class), I_r is between 0.67 and 1.0. If the network experienced the maximum expected overload (400% increase in load), a 267% to 400% increase in CS capacity would be required. A 150% increase in load would require CS capacity to at least be doubled. It can then generally be said that a CS policy would require at least double the capacity of a UL policy for the same weighted blocking performance.

C. Comparison of CS and UL on Implementation Time Scales

Another potentially more important issue in the comparison of CS and UL policies is the time scales on which they can be implemented. If capabilities for a UL policy are already installed in network hardware, a UL policy can be implemented within minutes of a major overload. The only delay would be to have time to assess new load levels. For the CS policy, however, to deploy new capacity when it was needed, it would take at least several hours and possibly days for this to occur. The use of a CS policy would result in extremely high blocking for several hours at the beginning of a major event when resources are needed most. Section VI provides an example that illustrates this. It should also be noted that the overloads that occur as a result of a major event typically are limited to the first day or two of an event [38]. If new CS capacity is not deployed soon enough, it might miss the overload period completely.

D. Knowledge of Resource Utilization

When using a CS policy, connections are admitted with no knowledge about the types of connections and the purposes for which they are being used. Network operators, therefore, are not able to know how their networks are utilized. In a detailed simulation of a disaster condition in [7], a link was considered that was loaded at 5 times its normal capacity. Of the offered load, 89.1% consisted of low-priority traffic. If a CS policy were used, 92.7% of the load admitted to the network would come from low-priority traffic and only 7.3% of the capacity would be used for high-priority traffic. Not only is the network overloaded, but the few resources that are available are being monopolized by low-priority users beyond the knowledge or control of the network operator.

If all high-priority traffic was admitted, however, 53.5% of the available capacity could be used for high-priority users. With an upper limit policy, along with the ticket server architecture in [8], network operators would be aware of the balance between classes and could control the way classes are defined and allocate resources as necessary. For the above scenario, a UL policy could have been implemented to give high-priority classes access to the 53.5% of the capacity that they needed with blocking probabilities of approximately zero.

E. Summary of the Comparison of CS and UL

The following summarizes the comparison of UL and CS polices for practical nonasymptotic conditions.

1) CS policies use considerably more capacity than UL policies. Extra capacity must be deployed at a rate 0.67 to 1.0 times the amount of the extra load from the disaster.
2) Installation of new capacity takes much longer than instituting an upper limit policy.
3) Typical load surges for a disaster last one or two days, so if installation of new capacity takes too long, it may not provide any benefit during the peak loading periods.
4) CS policies provide no knowledge about the use of resources. UL policies provide resource managers with extensive knowledge and control capabilities.

VI. Example

To illustrate the optimal upper limit policy methods proposed here, consider an example that closely replicates the situation that occurred during relief efforts for the Alfred P. Murrah Federal Building Bombing in Oklahoma City in 1995 [39]. The bombing occurred shortly after 9:00 a.m. on Wednesday, April 19, 1995. Immediately, serious congestion occurred on cellular telephone service provided by AT&T Wireless Services [39, p. 361]. To alleviate the problem, cellular system capacity was segmented into priority and nonpriority services according to a CP policy. Priority services were given to the fire department and other agencies through the use of special telephones that could use the priority resources. All others without special telephones could not obtain access to these priority channels. This prioritization mechanism was in place within 90 minutes of the event.

This prioritization of the cellular system was effective at reducing blocking for high-priority users that had the special equipment, but blocking for all others was made higher by partitioning. After nine hours at 6:00 p.m., a Cell on Wheels (COW) was installed in an attempt to alleviate congestion. This new capacity was still not enough, however. A second COW was installed the second day by 6:00 p.m., 33 hours after the event, which was able to add enough to the total capacity to provide sufficiently low blocking for all users.

If an upper limit policy had been used, several benefits would have been realized. First, if an upper limit policy and the ticket server architecture had been implemented, lower blocking would have been possible to all users within minutes, rather than having to wait 90 minutes. Second, all users would have been able to gain priority access to resources, not just those with special equipment. Instead of being given special equipment, priority users would just have obtained priority tickets. Third, some priority users did not have special equipment, either because the equipment had not been distributed yet or because they were not even considered for distribution of the equipment.

With a UL policy, all users are assumed to have equipment capable of contacting ticket servers to gain priority access. Users could be grouped into high priority and low-priority classes, then upper limits for each class could be set, which would reduce the need for installation of new capacity. It might still be desirable to install some new capacity to carry the nonpriority load, but it would not be as necessary.

Table II shows user classes for a scenario that reflects this example, making reasonable assumptions when exact data was not provided in [39]. Loads are shown for three classes during

TABLE II
TRAFFIC CLASSES FOR CELLULAR SYSTEM EXAMPLE

Traffic Class	Normal Load	Load During Disaster Relief
Class 1: Users with Special Equipment (High Priority)	15	60
Class 2: Users without Special Equipment – High Priority	10	40
Class 3: Users without Special Equipment – Low Priority	15	60

TABLE III
BLOCKING PROBABILITIES FOR CELLULAR SYSTEM EXAMPLE

Time Interval (Hours)	CS B_1 Class 1	UL B_1 Class 1	CS B_2 Class 2	UL B_2 Class 2	CS B_3 Class 3	UL B_3 Class 3
Before Event	$1x10^{-8}$	N/A	$1x10^{-8}$	N/A	$1x10^{-8}$	N/A
0-0.2	0.27	0.27	0.27	0.27	0.27	0.27
0.2-1.5	0.27	~0	0.27	~0	0.27	0.67
1.5-9	0.02	~0	0.51	~0	0.51	0.67
9-33	0.02	~0	0.23	~0	0.23	0.17
33+	0.02	~0	0.03	~0	0.03	0.17

normal conditions and during disaster recovery efforts. It is assumed that the disaster recovery load levels remain constant throughout the recovery period.

Using the timeline that actually occurred [39], Table III shows the CS blocking probabilities experienced by each class during the various time periods. Normal capacity is assumed to be 120 channels. When capacity is partitioned after 1.5 hours, high-priority traffic is given 70 channels and low-priority traffic is given 50 channels. These 50 channels expand to 80 and 110 channels with the first and second COWs.

When the priority mechanism is initiated at 1.5 hours, blocking for class 1 goes down from 0.27 to 0.02, while blocking for all others goes up from 0.27 to 0.51. The first COW at the ninth hour only reduces blocking to 0.23, but the second COW at the 33rd hour reduces blocking to 0.03. A second COW is necessary since priority users were included in the group of users who did not have special equipment.

Table III also shows what would happen if an upper limit policy were implemented. To respond quickly to the disaster, in ten minutes, a load estimate could be used to assign new upper limit thresholds to set blocking for the two priority classes to approximately zero. Over the next 60 to 90 minutes, the estimates could be refined as more arrivals occur. By 90 minutes, the lowest priority class would have blocking of 0.67. If a COW is installed to reduce the blocking for the lowest priority class, it could reduce blocking to 0.17. Use of a second COW might not be necessary, since blocking would be approximately 0 for high-priority classes and 0.17 for low-priority users.

An arrival rate estimate $\hat{\lambda}$ can be found by computing $\overline{T}$ as the average of N interarrival times seen from requests at the ticket server [8]. Assuming arrival times are stationary and exponentially distributed, $\overline{T}$ would be an N-stage Erlang random variable with mean $1/\lambda$ and standard deviation of $1/(\lambda\sqrt{N})$. Then $\hat{\lambda}_{\text{high}}$ could be set to $1/(\gamma\overline{T})$, $\gamma < 1$, which would overestimate the arrival rate so network operators could have high confidence that low blocking could be provided to priority users (at the expense of upper limits for other classes being set a little too low). From Table II, which is based on [39], and assuming average call holding times of five minutes, in ten minutes approximately 100 arrivals would occur per class. Using $\gamma = 0.885$ would provide 90% confidence that the actual arrival rate was less than $\hat{\lambda}_{\text{high}}$.

The results that most clearly highlight the benefit of the UL policy are for users from class 2. It is not until the 33rd hour of the event that these high-priority users receive blocking near zero using a CS policy; blocking near zero is provided within minutes with a UL policy. Clearly, a UL policy coupled with the ticket server architecture could have a dramatic effect in this type of disaster situation. Scenarios are also developed in [7] that show the benefits of a UL policy for multimedia connections on broadband landline networks.

VII. SUMMARY

This work showed that the benefits of prioritized resource allocation can be realized using simple algorithms. The paper presented a new UL policy methodology that optimized UL thresholds to provide preferred connection admission to high-priority traffic classes based on a weighted sum of blocking metric which had not been used before. The UL policy had already been found to have many advantages, especially when trying to explicitly control blocking. Here a new optimization formulation for upper limit policies was derived from Kelly's approximation for asymptotically large networks [31]. The result was a simple linear program and a simple algorithm that finds an optical CP policy and then uses a UL policy for practical implementation for an arbitrarily large network with an arbitrarily large number of classes.

This paper was the first to compare the amount of capacity needed to implement resource policies and their sensitivity to load variations. The upper limit policy was demonstrated to use less than half of the resources of complete sharing to provide comparable weighted blocking during typical disaster overload conditions. The UL policy was also demonstrated to be less sensitive than complete partitioning to the large load variations that can occur in high-priority traffic. When implemented along with the ticket server architecture in [7], public networks will be able to give preferred access to resources so that the important needs of society can be addressed when disasters or other special needs arise.

APPENDIX A

Overloaded conditions have been defined [32] as

$$\sum_{r=1}^{R} b_r \lambda_r > C. \tag{A-1}$$

When mean holding times are equal to one, the most likely state asymptotically, $\overline{\mathbf{n}}$, for CS policies is always on the constraint boundary

$$\sum_{r=1}^{R} b_r n_r = C. \tag{A-2}$$

This can be shown by using Kelly's equation in (3) with the constraint

$$\sum_{r=1}^{R} b_r n_r \leq C \tag{A-3}$$

which determines the structure of $\mathbf{A}$, resulting in

$$\begin{aligned} \min \quad & \sum_{r=1}^{R} \lambda_r e^{-y b_r} + Cy \\ \text{subject to} \quad & y \geq 0. \end{aligned} \tag{A-4}$$

The variable y is the Lagrange multiplier from a nonlinear optimization problem for finding the most likely state. Kuhn–Tucker conditions [40, page 314] can be used to determine if the constraint is active at the optimal point (i.e., the most likely state lies on the constraint boundary). If $y = 0$, the constraint need not be active. If $y = 0$, however, the optimal solution becomes

$$\sum_{r=1}^{R} b_r \lambda_r = C. \tag{A-5}$$

This result violates the definition of overloaded conditions from (A-1); therefore, $y > 0$ must be true and the constraint is necessarily active. Hence, the most likely state, $\overline{\mathbf{n}}$, for the CS policy must be on the constraint border for asymptotically large networks in overloaded conditions.

Using a similar approach, conditions can be derived where the MLS for a CP policy is also on the boundary of the CS region (i.e., on the "corner" of the CP region). It would require all CP constraints to be active, necessitating $y_r > 0$ for all R constraints from (9), so $b_r \lambda_r > C_r$ for all r.

Appendix B

This Appendix shows that if two particular states within region A of Fig. 2(b) are less likely than the MLS of the CP policy, then all states within region A are less likely. Some or all of them can be added to the CP region to create a new policy with a region that has the same MLS.

First of all, consider the following state which corresponds to adding one class 1 connection to the MLS (assuming $b_2 = 1$ and $b_1 > 1$).

$$\mathbf{n}_{\mathbf{A}}^{(1)} = (\overline{n}_1 + 1, \overline{n}_2 - b_1). \tag{B-1}$$

If the following ratio using (6) of its state probability to the state probability of the MLS at $\overline{\mathbf{n}}$ is less than 1

$$\begin{aligned} \frac{\pi\left(\mathbf{n}_{\mathbf{A}}^{(1)}\right)}{\pi(\overline{\mathbf{n}})} &= \left(\frac{\lambda_1^{\overline{n}_1+1}\lambda_2^{\overline{n}_2-b_1}}{\lambda_1^{\overline{n}_1}\lambda_2^{\overline{n}_2}}\right)\left(\frac{\overline{n}_1!\overline{n}_2!}{(\overline{n}_1+1)!(\overline{n}_2-b_1)!}\right) \\ &= \frac{\lambda_1 \prod_{i=1}^{b_1}(\overline{n}_2 - b_1 + i)}{\lambda_2^{b_1}(\overline{n}_1+1)} \end{aligned} \tag{B-2}$$

then this state is less likely than the MLS. Now if we consider the following state

$$\mathbf{n}'_{\mathbf{A}} = (\overline{n}_1 + k_1 + 1, \overline{n}_2 - (k_1+1)b_1 - k_2) \tag{B-3}$$

which is the general state in region A, $\mathbf{n}_{\mathbf{A}}$ from (18), with one more class 1 connection, then we find the ratio

$$\begin{aligned} \frac{\pi(\mathbf{n}'_{\mathbf{A}})}{\pi(\mathbf{n}_{\mathbf{A}})} &= \left(\frac{\lambda_1^{\overline{n}_1+k_1+1}\lambda_2^{\overline{n}_2-(k_1+1)b_1-k_2}}{\lambda_1^{\overline{n}_1+k_1}\lambda_2^{\overline{n}_2-k_1b_1-k_2}}\right) \\ &\quad \times \left(\frac{(\overline{n}_1+k_1)!(\overline{n}_2-k_1b_1-k_2)!}{(\overline{n}_1+k_1+1)!(\overline{n}_2-(k_1+1)b_1-k_2)!}\right) \\ &= \frac{\lambda_1 \prod_{i=1}^{b_1}(\overline{n}_2 - b_1 + i - k_1b_1 - k_2)}{\lambda_2^{b_1}(\overline{n}_1+k_1+1)}. \end{aligned} \tag{B-4}$$

This ratio is always less than (B-2), since the numerator of (B-4) is smaller than the numerator in (B-2), and the denominator is larger for all k_1 greater than 0 and k_2 greater than or equal to 0. Therefore, every state is less likely than the state in (B-1) if (B-2) is less than 1.

Second, consider the following state which corresponds to removing one class 2 connection from the MLS

$$\mathbf{n}_{\mathbf{A}}^{(2)} = (\overline{n}_1, \overline{n}_2 - 1). \tag{B-5}$$

If the following ratio using (6) of its state probability to the state probability of the MLS at $\overline{\mathbf{n}}$ is less than 1

$$\begin{aligned} \frac{\pi\left(\mathbf{n}_{\mathbf{A}}^{(2)}\right)}{\pi(\overline{\mathbf{n}})} &= \left(\frac{\lambda_1^{\overline{n}_1}\lambda_2^{\overline{n}_2-1}}{\lambda_1^{\overline{n}_1}\lambda_2^{\overline{n}_2}}\right)\left(\frac{\overline{n}_1!\overline{n}_2!}{\overline{n}_1!(\overline{n}_2-1)!}\right) \\ &= \frac{\overline{n}_2}{\lambda_2}, \end{aligned} \tag{B-6}$$

then this state is less likely than the MLS. Now if we consider the following state

$$\mathbf{n}''_{\mathbf{A}} = (\overline{n}_1 + k_1, \overline{n}_2 - k_1b_1 - k_2 - 1). \tag{B-7}$$

which is the general state in region A, $\mathbf{n}_{\mathbf{A}}$ from (18), with one less class 2 connection, then we find the ratio

$$\begin{aligned} \frac{\pi(\mathbf{n}''_{\mathbf{A}})}{\pi(\mathbf{n}_{\mathbf{A}})} &= \left(\frac{\lambda_1^{\overline{n}_1+k_1}\lambda_2^{\overline{n}_2-k_1b_1-k_2-1}}{\lambda_1^{\overline{n}_1+k_1}\lambda_2^{\overline{n}_2-k_1b_1-k_2}}\right) \\ &\quad \times \left(\frac{(\overline{n}_1+k_1)!(\overline{n}_2-k_1b_1-k_2)!}{(\overline{n}_1+k_1)!(\overline{n}_2-k_1b_1-k_2-1)!}\right) \\ &= \frac{\overline{n}_2 - k_1b_1 - k_2}{\lambda_2}. \end{aligned} \tag{B-8}$$

This ratio is always less than (B-6), since the numerator is smaller than the numerator in (B-6) for all k_1 greater than or equal to 0 and k_2 greater than 0. Therefore, every state is less likely than the state in (B-5) if (B-6) is less than 1.

In conclusion, if (B-2) and (B-6) are less than one, then states $\mathbf{n}_{\mathbf{A}}^{(1)}$ and $\mathbf{n}_{\mathbf{A}}^{(2)}$ are less likely than the MLS and all other states with more class 1 connections and/or less class 2 connections are less likely than $\mathbf{n}_{\mathbf{A}}^{(1)}$ and $\mathbf{n}_{\mathbf{A}}^{(2)}$. Some or all states can be included in another policy that will have the same MLS and, hence, the same asymptotic blocking.

Appendix C

This section provides details on removing limits of a CP policy to convert to a UL policy of R classes of users. The

basic approach is to find the most likely state from the optimal complete partitioning policy, then remove one constraint, C_k, and see if a change occurs in the most likely state. The form of the constraint matrices then becomes the original matrix $\mathbf{A}$ with a row removed for constraint C_k, and a new row added for the overall capacity constraint. By using equation (3), using y_r to denote the Lagrange multiplier for each C_r constraint, and y to denote the Lagrange multiplier for the capacity constraint,

$$\min S(y, y_r) = \min\left[\sum_{\substack{r=1\\ r\neq k}}^{R} \lambda_r e^{-b_r y_r - y b_r} + \lambda_k e^{-y b_k} + \sum_{\substack{r=1\\ r\neq k}}^{R} C_r y_r + yC\right] \tag{C-1}$$

subject to $y_r \geq 0$, $y \geq 0$.

The partial derivative with respect to y_r is

$$\frac{\partial S(y, y_r)}{\partial y_r} = -b_r \lambda_r e^{-b_r y_r - y b_r} + C_r = 0. \tag{C-2}$$

The partial derivative with respect to y is

$$\frac{\partial S(y, y_r)}{\partial y} = -\sum_{\substack{r=1\\ r\neq k}}^{R} b_r \lambda_r e^{-b_r y_r - y b_r} + C - b_k \lambda_k e^{-y b_k} = 0. \tag{C-3}$$

Putting the results of (C-2) into (C-3)

$$-\sum_{\substack{r=1\\ r\neq k}}^{R} C_r + C - b_k \lambda_k e^{-y b_k} = 0$$

$$e^{-y b_k} = \frac{C_k}{b_k \lambda_k}. \tag{C-4}$$

To meet the requirement $y \geq 0$, $C_k \leq b_k \lambda_k$ must be true. This is a constraint of the problem for optimizing weighted blocking in (13), so this condition will be true for all k.

Putting the result of (C-4) back into (C-2)

$$b_r \lambda_r e^{-b_r y_r} \left(\frac{C_k}{b_k \lambda_k}\right)^{b_r/b_k} = C_r$$

$$e^{-b_r y_r} = \frac{C_r}{b_r \lambda_r} \left(\frac{C_k}{b_k \lambda_k}\right)^{-b_r/b_k}. \tag{C-5}$$

To have $y_r \geq 0$

$$\frac{C_r}{b_r \lambda_r} \left(\frac{C_k}{b_k \lambda_k}\right)^{-b_r/b_k} \leq 1 \qquad \forall r \neq k, r = 1, \ldots, R. \tag{C-6}$$

So, for class k to have its constraint removed from the UL policy, (C-6) must be true for all r not equal to k. A search can be conducted over all possible combinations of k and r to remove UL constraints that meet (C-6). These constraints can then be removed, since no changes in the most likely state would result.

One simplification can be made to (C-6), however. For classes k where $C_k = b_k \lambda_k$, the conditions of (C-6) will automatically be met, since $C_r \leq b_r \lambda_r$ will be true for all r. This corresponds to a class k where the CP optimization process has sought to make blocking from (10)

$$B_k = 1 - \frac{C_k}{b_k \lambda_k} = 0. \tag{C-7}$$

Therefore, no upper limit need be imposed on any class where blocking was intended to be zero from the optimization process.

REFERENCES

[1] Federal Emergency Management Agency, "Technology applications by the federal emergency management agency in response, recovery, and mitigation operations," presented at the 27th Joint Meeting of the U.S./Japanese Panel on Wind and Seismic Effects, Tokyo/Osaka, Japan, May 16–27, 1995.

[2] G. Philip and R. Hodge, "Disaster area architecture," in *Proc. IEEE MILCOM*, 1995, pp. 833–837.

[3] S. Adamson and S. Gordon, "Analysis of two trunk congestion relief schemes," in *Proc. IEEE MILCOM*, 1993, pp. 902–906.

[4] Computer Science and Telecommunications Board (CSTB), National Research Council, *Computing and Communications in the Extreme: Research for Crisis Management and Other Applications*. Washington, DC: National Academy Press, 1996.

[5] International Telecommunication Union, Telecommunication Standardization Sector, "International Emergency Preparedness Scheme," ITU Recommendation, E.106, Mar. 2000.

[6] K. Carlberg and I. Brown, "Framework for supporting IEPS in IP telephony," Internet Engineering Task Force Internet Draft, Work in Progress, Nov. 2000.

[7] C. Beard, "Dynamic agent based prioritized resource allocation for stressed networks," Ph.D. dissertation, Univ. of Kansas, Lawrence, 1999.

[8] C. Beard and V. Frost, "Dynamic agent-based prioritized connection admission for stressed networks," in *1999 IEEE Int. Conf. Communications*, Vancouver, B.C., Canada, June 1999.

[9] ——, "Ticket server performance evaluation using a hybrid simulation approach," in *Proc. Soc. Computer Simulation 2001 Advanced Simulation Technologies Conf.*, Seattle, WA, Apr. 2001, pp. 74–81.

[10] J. S. Kaufman, "Blocking in a shared resource environment," *IEEE Trans. Commun.*, vol. COM-29, pp. 1474–1481, Oct. 1981.

[11] G. Foschini, B. Gopinath, and J. Hayes, "Optimum allocation of servers to two types of competing customers," *IEEE Trans. Commun.*, vol. COM-29, pp. 1051–1055, July 1981.

[12] P. B. Key, "Optimal control and trunk reservation in loss networks," *Probabil. Eng. Inform. Sci.*, vol. 4, pp. 203–242, 1990.

[13] R. Rajan *et al.*, "A policy framework for integrated and differentiated services in the Internet," *IEEE Network Mag.*, pp. 36–41, Sept./Oct. 1999.

[14] K. W. Ross, *Multiservice Loss Models for Broadband Telecommunication Networks*. London, U.K.: Springer-Verlag, 1995.

[15] K. Ross and D. Tsang, "The stochastic knapsack problem," *IEEE Trans. Commun.*, vol. 37, pp. 740–747, July 1989.

[16] B. Kraimeche and M. Schwartz, "Circuit access control strategies in integrated digital networks," in *IEEE INFOCOM*, San Francisco, CA, Apr. 1984, pp. 230–235.

[17] I. Gopal and T. Stern, "Optimal call blocking policies in an integrated services environment," in *Proc. 17th Conf. Information Science and Systems*. Baltimore, MD: The Johns Hopkins Univ., 1983, pp. 383–388.

[18] S. Jordan and P. Varaiya, "Control of multiple service, multiple resource communication networks," *IEEE Trans. Commun.*, vol. 42, pp. 2979–2988, Nov. 1994.

[19] J. Hyman, A. Lazar, and G. Pacifici, "A separation principle between scheduling and admission control for broadband switching," *IEEE J. Select. Areas Commun.*, vol. 11, pp. 605–616, May 1993.

[20] G. Choudhury, K. Leung, and W. Whitt, "An Algorithm to compute blocking probabilities in multi-rate multi-class multi-resource loss models," *Advances in Applied Probability*, vol. 27, pp. 1104–1143, 1995.

[21] ——, "Efficiently providing multiple grades of service with protection against overloads in shared resources," *AT&T Tech. J.*, pp. 50–63, July/Aug. 1995.

[22] S. K. Biswas and B. Sengupta, "Call admissibility for multirate traffic in wireless ATM networks," in *Proc. IEEE INFOCOM*, vol. 2, Kobe, Japan, Apr. 1997, pp. 649–657.

[23] S. C. Borst and D. Mitra, "Virtual partitioning for robust resource sharing: Computational techniques for heterogeneous traffic," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 668–678, June 1998.
[24] A. K. Choudhury and E. L. Hahne, "Dynamic queue length thresholds in a shared memory ATM switch," in *Proc. IEEE INFOCOM*, vol. 2, Mar. 1996, pp. 679–687.
[25] S. Krishnan, A. K. Choudhury, and F. M. Chiussi, "Dynamic partitioning: A mechanism for shared-memory management," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 144–152.
[26] K. Kumaran and D. Mitra, "Performance and fluid simulations of a novel shared buffer management scheme," in *Proc. IEEE INFOCOM*, Mar. 1998, pp. 1449–1461.
[27] A. Berger and W. Whitt, "Extending the effective bandwidth concept to networks with priority classes," *IEEE Commun. Mag.*, vol. 36, pp. 78–83, Aug. 1998.
[28] A. Elwalid, D. Mitra, and R. H. Wentworth, "A new approach for allocating buffers and bandwidth to heterogeneous, regulated traffic in an ATM node," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1115–1127, Aug. 1995.
[29] A. Berger and W. Whitt, "Effective bandwidths with priorities," *IEEE/ACM Trans. Networking*, vol. 6, pp. 447–460, Aug. 1998.
[30] I. Zeng and C. Beard, "Prioritized resource allocation at differentiated services ingress routers," in *Proc. IASTED Parallel and Distributed Systems Conf.*, Las Vegas, NV, Nov. 2000, pp. 508–513.
[31] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," *Adv. Appl. Probl.*, vol. 18, pp. 473–505, 1986.
[32] P. J. Hunt and F. P. Kelly, "On critically loaded loss networks," *Adv. Appl. Probl.*, vol. 21, pp. 831–841, 1989.
[33] M. I. Reiman, "A critically loaded multiclass Erlang loss system," *Queueing Syst.*, vol. 9, pp. 65–82, 1991.
[34] ——, "Some allocation problems for critically loaded loss systems with independent links," *Perform. Eval.*, vol. 13, pp. 17–25, 1991.
[35] A. A. Puhalskii and M. I. Reiman, "A critically loaded multirate link with trunk reservation," *Queueing Syst.*, vol. 28, pp. 157–190, 1998.
[36] N. G. Bean, R. J. Gibbens, and S. Zachary, "Asymptotic analysis of single resource loss systems in heavy traffic, with applications to integrated networks," *Adv. Appl. Probl.*, vol. 27, pp. 273–292, 1995.
[37] FCC Operational Requirements Subcommittee, "Final report," presented to Public Safety Wireless Advisory Committee, National Telecommunications and Information Agency, Washington, DC, May 29, 1996.
[38] Focus Group 5, "Network reliability—The path forward: Telecommuting as a back-up in emergencies," Network Reliability Council, Feb. 1996.
[39] The City of Oklahoma City, *Alfred P. Murrah Federal Building Bombing: Final Report*. Stillwater, OK: Fire Protection Publications, 1996.
[40] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

Cory C. Beard (S'88–M'99) received the B.S. and M.S. degrees from the University of Missouri, Columbia, in 1990 and 1992, respectively, and the Ph.D. degree from the University of Kansas, Lawrence, in 1999.

He is currently an Assistant Professor in the Computer Networking Cluster of the School of Interdisciplinary Computing and Engineering at the University of Missouri, Kansas City. He was a Project Electrical Engineer for the Transmission and Distribution Division of Black & Veatch Consulting Engineers from 1992 to 1995. His current research interest is in the areas of prioritized and differentiated quality of service provisioning, network traffic engineering, queueing theory, intelligent agents, and the application of computer networks to disaster recovery operations.

Dr. Beard was a recipient of the National Science Foundation Graduate Fellowship and the Madison and Lila Self Graduate Fellowship at the University of Kansas.

Victor S. Frost (S'75–M'82–SM'90–F'98) received the B.S., M.S., and Ph.D. degrees from the University of Kansas, Lawrence, in 1977, 1978, and 1982, respectively.

He is currently the Dan F. Servey Distinguished Professor of Electrical Engineering and Computer Science and Director of the University of Kansas Information and Telecommunications Technology Center. From 1987 to 1996, he was the Director of the Telecommunications and Information Sciences Laboratory. He is a member of the Senator Pat Roberts Task Force on Information, Telecommunications and Computing Technology. He was Principal Investigator on the University of Kansas MAGIC gigabit network research effort and ACTS ATM Internetwork (AAI). His research has been sponsored by NSF, DARPA, Rome Labs, and NASA, Sprint, NEC America, NCR, BNR, NEC, Telesat Canada, AT&T, McDonnel Douglas, and COMDISCO Systems. His current research interest is in the areas of integrated broadband communication networks, communications system analysis, and traffic and network simulation.

Dr. Frost received the Presidential Young Investigator Award from the National Science Foundation in 1984. He received an Air Force Summer Faculty Fellowship, a Ralph R. Teetor Educational Award from the Society of Automotive Engineers, and the Miller Professional Development Awards for Engineering Research and Service in 1986 and 1991, respectively.