# Resource Stealing in Endpoint Controlled Multi-class Networks ⋆

Susana Sargento[1], Rui Valadas[1], and Edward Knightly[2]

[1] Institute of Telecommunications, University of Aveiro, Portugal
[2] ECE Department, Rice University, USA

**Abstract.** Endpoint admission control is a mechanism for achieving scalable services by pushing quality-of-service functionality to end hosts. In particular, hosts probe the network for available service and are admitted or rejected by the host itself according to the performance of the probes. While particular algorithms have been successfully developed to provide a single service, a fundamental *resource stealing* problem is encountered in multi-class systems. In particular, if the core network provides even rudimentary differentiation in packet forwarding (such as multiple priority levels in a strict priority scheduler), probing flows may infer that the quality-of-service in their own priority level is satisfactory, but may inadvertently and adversely affect the performance of other classes, stealing resources and forcing them into quality-of-service violations. This issue is closely linked to the network scheduler as the performance isolation property provided by multi-class schedulers also introduces limits on *observability*, or a flow's ability to assess its impact on other traffic classes. In this paper, we study the problem of resource stealing in multi-class networks with end-point probing. For this scalable architecture, we describe the challenge of simultaneously achieving multiple service levels, high utilization, and a strong service model without stealing. We propose a probing algorithm termed $\varepsilon$-probing which enables observation of other traffic classes' performance with minimal additional overhead. We next develop a simple but illustrative Markov model to characterize the behavior of a number of schedulers and network elements, including flow-based fair queueing, class-based weighted fair queueing and rate limiters. Finally, we perform an extensive set of simulation experiments to study the performance tradeoffs of such architectures, and to evaluate the effectiveness of $\varepsilon$-probing.

## 1 Introduction

The Integrated Services (IntServ) architecture of the IETF provides a mechanism for supporting quality-of-service for real-time flows. Two important components of this architecture are admission control [3,8] and signaling [14]: the former ensures that sufficient network resources are available for each new flow, and the latter communicates such resource demands to each router along the flow's path. However, the demand for high-speed core routers to process per-flow reservation requests introduces scalability and deployability limitations of this architecture without further enhancements.

⋆ Susana Sargento is supported by Ph.D. scholarship PRAXIS XXI/BD/15678/98. Edward Knightly is supported by NSF CAREER Award ANI-9733610, NSF Grants ANI-9730104 and ANI-0085842, a Sloan Fellowship, and Texas Instruments.

S. Palazzo (Ed.): IWDC 2001, LNCS 2170, pp. 195–211, 2001.

In contrast, the Differentiated Services (DiffServ) architecture [2,9] achieves scalability by limiting quality-of-service functionalities to class-based priority mechanisms together with service level agreements. However, without per-flow admission control, such an approach necessarily weakens the service model as compared to IntServ, namely individual flows are not assured of a bandwidth or loss guarantee.

A key challenge addressed in recent research is how to simultaneously achieve the scalability of DiffServ and the strong service model of IntServ. Towards this end, several novel architectures and algorithms have been proposed. For example, architectures for scalable deterministic services were developed in [12,15]. In [12], a technique termed Dynamic Packet State is developed in which inserting state information into packet headers overcomes the need for per-flow signaling and state management. In [15], a bandwidth broker is employed to manage deterministic services without explicit coordination among core nodes. A scheme that provides scalable *statistical* services is developed in [5], whereby only a flow's egress node performs admission control via continuous passive monitoring of the available service on a path.

While such approaches are able to achieve scalability and strong service models, they do so while requiring specific functionality to be employed at edge and/or core nodes. For example, [5] requires packet time stamping and egress nodes to process signaling messages; [12] requires rate monitoring and state packet insertion at ingress points and special schedulers at core nodes. Thus, despite that such edge/core router modifications may indeed be feasible, an alternate and equally compelling problem is to ask whether the same goals can be achieved without *any* changes to core or edge routers, or at most with routers providing simplistic prioritized forwarding as envisioned by DiffServ extensions such as class based queueing or prioritized dropping policies.

This design constraint is quite severe: it precludes use of a signaling protocol as well as any special packet processing within core nodes. Such a constraint naturally leads to probing schemes in which end hosts perform admission control by assessing the state of the network by transmitting a sequence of probe packets and measuring the corresponding performance. If the performance (e.g., loss rate) of the probes is acceptable, the flow is admitted, otherwise it is rejected. Design and analysis of several such schemes can be found in [1,6,7]. Such approaches achieve scalability by pushing quality-of-service functionalities to the end system and indeed removing the need for a signaling protocol or any special-purpose edge or core router functions. Moreover, [4] found that such an architecture is indeed able to provide a single controlled-load like service as defined in [13].

However, can host-controlled probing schemes be generalized to support *multiple* service classes as achieved by both IntServ and DiffServ? In particular, DiffServ supports multiple service classes differentiated by simple aggregate scheduling policies (per-hop behaviors); DiffServ's Service Level Agreements (SLAs) provide aggregate bandwidth guarantees to traffic classes; IntServ provides mechanisms to associate different quality-of-service parameters (e.g., loss rate, bandwidth, and delay) with different traffic classes. Can such multi-class service models co-exist with the host-controlled architecture?

Unfortunately, a resource *stealing* problem, first described in [4], can occur in multi-class systems. In particular, the problem occurs when a user probes within its desired class and, upon obtaining no loss (or loss below the class' threshold), infers that sufficient

capacity is available, which indeed it may be within the class. However, in some cases, admission of the new probing flow would force *other* classes into a situation of quality-of-service violations, unbeknownst to the probing flow. Such resource stealing, described in detail in Section 2, arises from a fundamental observability issue in a multi-class system: the performance isolation property provided by multi-class networks also inhibits flows from assessing their performance impact on other classes.

The goal of this paper is to investigate host probing in multi-class networks. Addressing the problem of resource stealing, our contribution is threefold. First, we study architectural issues and show how service disciplines and the work conservation property have important roles in the performance of probing systems. For example, while a non-work-conserving service discipline can prohibit resource borrowing across classes and remove the stealing problem, such rigid partitioning of system resources limits resource utilization. Second, we develop a probing algorithm which simultaneously achieves high utilization and a strong service model without stealing. The algorithm, termed $\varepsilon$-probing, provides a minimally invasive mechanism to enable flows to assess their impact on *other* traffic classes in Class-Based Fair Queueing (CBQ) and strict priority systems. Finally, we introduce a simple but illustrative analytical model based on Markov Chains. Using the model, we precisely identify stealing states, comparatively analyze several probing architectures, and quantify the aforementioned tradeoffs.

In all cases, we use an extensive set of simulations to evaluate different probing schemes and architectures under a wide range of scenarios and traffic types. The experimental results indicate that $\varepsilon$-probing can achieve utilizations close to the limits obtained by fair queueing, while *eliminating* resource stealing. Consequently, if core networks provide minimal differentiation on the forwarding path, $\varepsilon$-probing provides a scalable mechanism to control multiple service classes, achieve high utilization, and provide a strong service model without resource stealing.

The remainder of this paper is organized as follows. In Section 2, we formulate the stealing problem in multi-class networks and describe the role of the packet scheduler. Next, in Section 3, we propose a simple probing algorithm, termed $\varepsilon$-probing, that overcomes the observability limitations introduced by multi-class schedulers. In Section 4 we develop an analytical model to study the performance issues and tradeoffs in achieving high utilization, multiple service classes, and a strong service model without stealing. Finally, in Section 5, we describe an extensive set of simulation experiments used to investigate the design space under more realistic scenarios.

## 2 Resource Stealing

The stealing problem arises in multi-class systems in which resources are remotely controlled by observation. This is in contrast to systems in which resources are controlled with explicit knowledge of their load, such as in IntServ-like architectures. In this section, we describe the origins of multi-class stealing and the corresponding design and performance issues. Throughout, we consider a general definition of “class” that can be based on application types, service level agreements, etc., and with quality-of-service parameters such as loss rate and delay associated with each class.

### 2.1 Origins and Illustration

Probing schemes, such as those studied in [1,4,6,7], can be described with an example using the network depicted in Figure 1. To establish a real-time flow between hosts H and H', host H transmits a sequence of probes into the network at the desired rate (or peak rate for variable rate flows). If the loss rate of the probes is below a pre-established threshold for the traffic class, then the flow is admitted, and otherwise it is rejected. Scalability is achieved in such a framework by pushing all quality-of-service functionality to end-hosts, indeed removing the need for *any* signaling or storage of per-flow state.

The stealing problem can be illustrated as follows. Consider the following simple scenario with two flows sharing a single router with link capacity $C$ and a flow-based fair queueing scheduler[1](or similarly core stateless fair queueing [11] to achieve scalability on the data path). Suppose that the first flow requires a bandwidth of $\frac{3}{4}C$ and is admitted to an initially idle system. Further suppose that the second flow has a bandwidth requirement $\frac{1}{2}C$. Upon probing for the available service in the fair queueing system, the flow will discover that it can indeed achieve a loss-free service with throughput $\frac{1}{2}C$, and admit itself. Unfortunately, while $\frac{1}{2}C$ is indeed the fair rate for each flow, the goal here is not to achieve packet-level fairness, but rather to achieve flow-level quality-of-service objectives. Thus, in this example, abruptly reducing the first flow's capacity is a clear violation of the flow's service.

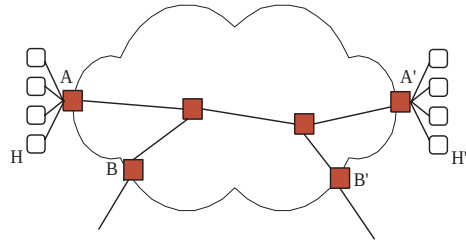



**Fig. 1.** Illustration of Probing and Multi-Class Stealing

This simple example illustrates an important point. The ability of fair queueing to provide performance isolation can be exploited for both flow-control (to quickly and accurately assess a flow's fair rate) and quality of service (to provide a minimum guaranteed bandwidth to a flow or group of flows). However, it is precisely this performance isolation which introduces the "stealing" problem for scalable services: since the probing flow is isolated from the established flows, it cannot assess the potentially significant performance impact that it has on them. Consequently, while a new flow can determine whether or not its own quality-of-service objectives will be satisfied, it cannot determine

[1] We will discuss both flow- and class-based fair queueing. In class-based, the scheduling discipline inside each class is FCFS (First Come First Served), and between classes the discipline is fair queueing. In flow based each flow is considered as a class.

its impact on others. Thus, if admitted, the new flow can unknowingly steal resources from previously admitted flows and result in service violations.

This problem is not limited to fair queueing nor to per-flow schedulers. Consider a class-based strict priority scheduler in which a new flow wishes to probe for the available service at a mid-level priority. Ideally, the flow could indeed assess the capacity remaining from higher priority flows by probing at the desired service level. However, it would not be able to assess its impact on lower priority levels without also probing at lower levels.

Thus, the stealing problem arises from a lack of *observability* of multi-class networks, namely, that assessing one's own performance does not necessarily ensure that other flows are not adversely affected.

### 2.2 Problem Formulation

Within a framework of scalable services based on host probing, the key challenge is to simultaneously achieve (1) multiple traffic classes (differentiated services), (2) high utilization, and (3) a strong service model without stealing. To illustrate this challenge, consider the network of Figure 1 in which each link has capacity $C$. Further suppose the system supports two traffic classes $\mathcal{A}$ and $\mathcal{B}$ with different traffic characteristics and QoS requirements.

A key design axis which affects these design goals is whether or not the system allows resource sharing across classes. This in turn is controlled by the scheduler and whether or not it is work conserving.

**Rigid Partitioning without Work Conservation** One way to ensure both classes achieve their desired QoS constraints is via hard partitioning of system resources with no bandwidth borrowing across classes allowed. Such a system can be implemented with rate limiters, i.e., policing elements with the peak rate of class $i$ limited to $\phi_i C$ with $\phi_1 + \phi_2 \leq 1$.

Observe that a hard partitioning system can support multiple traffic classes and does not incur stealing, thereby achieving the first and third goal above. However, notice that the system is non-work-conserving in the sense that it will reject flows even if sufficient capacity is available and consequently can under utilize system resources. For example, suppose path A-A' of Figure 1 has a large class-$\mathcal{A}$ demand and no class-$\mathcal{B}$ and vice versa on path B-B'. In this case, the system would be under-utilized as only half of the flows which the system could support would be admitted. In general, whenever the current bandwidth demands are not in line with the weights $\phi_i$, the system will suffer from low utilization.

**Inter-class Sharing with Work Conservation** In contrast to the scenario above, consider a work-conserving system which allows one class to use excess capacity from other classes. In particular, consider a two-class fair queueing system (without rate limiters) with weights $\phi_1$ and $\phi_2$. With the same demand as in the example above, both A-A' and B-B' flows can fully utilize the capacity due to the soft partitioning of resources in the work conserving system. Thus, the first and second goals are achieved. However, as described in Section **??**, such a system suffers from the stealing problem, as a new class-$\mathcal{B}$ flow on A-A' or a new class-$\mathcal{A}$ flow on B-B' will steal bandwidth from established flows.

**Targeted Behavior** The targeted behavior that we strive to achieve is to combine the advantages of the hard and soft partitioning systems and allow borrowing across classes to achieve high utilization, while eliminating resource stealing to provide a strong service model. Thus, in the example, if A-A' is fully utilized by class-$\mathcal{A}$ flows, class-$\mathcal{B}$ flows (and class-$\mathcal{A}$ flows) should be blocked until class-$\mathcal{A}$ flows depart. Below, we develop new probing schemes which seek to simultaneously achieve the above three design goals and achieve this targeted behavior. This service model is a greedy one, in which all flows which can be admitted are, provided that their and all other service requirements can be satisfied. This strategy does not incorporate blocking probability as a QoS parameter. It is possible to have targeting blocking probabilities, but it is beyond the scope of this paper. Throughout the paper we will only consider the admission controlled traffic. Best-effort would have a lower priority level so it would not interfere with the admission controlled one. Also, guaranteed-like service with strict QoS assurances would have a reserved bandwidth and a higher priority.

## 3 Epsilon Probing

In this section, we develop probing algorithms which overcome the stealing problem in fair queueing multi-class servers. The key technique is to infer the "state" of other classes with minimal overhead in terms of probing traffic or probing duration. Throughout, we consider a simplified bufferless fluid model as in [4], in which flows and probes transmit at constant rate and probing is "perfect" in the sense that probes correctly infer their loss rate as determined by the scheduling discipline (which defines how loss is distributed among classes) and the workload (which defines the extent of the loss in the system).

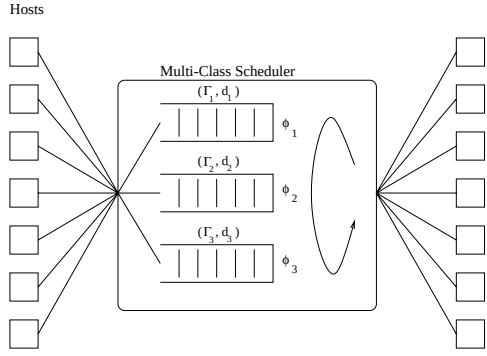



**Fig. 2.** Illustration of $\varepsilon$-Probing

Consider a class-based weighted fair queueing server with $K$ classes, where class $k$ has weight $\phi_k$ and target QoS parameters of loss rate $\Gamma_k$ and delay bound $d_k$ (for simplicity, we restrict the discussion to loss). According to the definition of WFQ, the bandwidth utilized by class $k$ when all classes are backlogged is given by $U_k = \frac{\phi_k}{\sum_{i \in \mathcal{S}} \phi_i} C$ where $\mathcal{S}$ is the set of backlogged classes. Let the demanded bandwidth of

class $k$ be denoted by $B_k$. If $B_1 + \cdots + B_K > C$, then loss occurs in the system, and the loss rate of class $k$ at that particular instant in time is given by

$$\gamma_k = (B_k - U_k)^+ / B_k. \tag{1}$$

With probing in a single level, a new class-$k$ flow requesting bandwidth $b_k$ is admitted if its measured loss rate is less than the class requirement, i.e., if $\gamma_k \leq \Gamma_k$. However, observe that even under congestion and arbitrarily high loss rates in other classes, the new flow would be admitted as long as

$$B_k + b_k \leq C\phi_k/(1 - \Gamma_k). \tag{2}$$

Thus, stealing across classes can occur as the probing flow fails to observe whether or not other classes' loss requirements are also satisfied. While simultaneously probing in all classes may seemingly solve the problem, it is not only unnecessary, but significantly damages the performance of the system: namely increased probing traffic forces the system to more quickly enter a thrashing regime in which excessive probing traffic causes flows to be mistakenly rejected, and in the limit, causes system collapse [4].

We propose $\varepsilon$-probing as a probing scheme designed to eliminate stealing in a minimally invasive way. With $\varepsilon$-probing, a new flow requesting bandwidth $b_k$ *simultaneously* transmits a small bandwidth $\varepsilon_i$ to each other class $i$. The motivating design principle is that the impact of the new flow on *all* classes must be observed, so that the new flow is only admitted if $\gamma_i \leq \Gamma_i$ is satisfied for all $i = 1, \cdots, K$. The admissible loss rate in each $\varepsilon$-probe ($\gamma_i$) is the same for all classes and globally agreed upon. In particular, addition of the new class-$k$ flows can affect $U_k$ for each class: the $\varepsilon$-probes ensure that the new $U_k$ is sufficiently large to meet the required loss rate.

In the fluid model, $\varepsilon_i$ can be arbitrarily small, whereas in the packet system, it must be sufficiently large to detect loss in the class. In the simulation experiments of Section 5, we consider $\varepsilon_i$ = 64 kb/sec for a 45 Mb/sec link with flows transmitting at rates between 512 kb/sec and 2 Mb/sec.

Finally, we note that despite the utilization advantages of a work-conserving system, a network may still contain non-work conserving elements to achieve other objectives (e.g., to ensure that a minimum bandwidth is always available in each class, even if there is no current demand, cf. Figure 7). The goal of $\varepsilon$-probing is to enable inter-class resource sharing to the maximal extent allowed by the system architecture.

$\varepsilon$-probing is applicable to both class-based fair queueing and strict priority schedulers. In the latter type of scheduler, the $\varepsilon$-probes are required only in the priority levels lower than the level of the class for the flow that is requesting admission. In higher levels stealing cannot occur and no $\varepsilon$-probe is required. Therefore, the overhead due to $\varepsilon$-probes is lower in strict priority than in class-based fair queueing schedulers.

## 4 Theoretical Model

In this section we develop an analytical model based on continuous time Markov chains to study the problem of resource stealing in multi-class networks.

### 4.1 Preliminaries

In the model, each state identifies the number of currently admitted flows in each class such that with $K$ classes, the Markov chain has $K$ dimensions. The link capacity is $C$ resource units and each flow of class $k$ occupies $b_k$ resource units.[2] We assume that new flows arrive to the system as a Poisson process with mean inter-arrival time $\frac{1}{\lambda_k}$, and that flow lifetimes are also exponentially distributed with mean $\frac{1}{\mu_k}$. Probing is considered instantaneous so that we do not consider the "thrashing" phenomenon (due to simultaneously probing flows) described in [4].

We define $n_k$ to be the number of class $k$ flows in the system such that the total amount of resources occupied by all flows in the system is given by $(\mathbf{b} \cdot \mathbf{n})$, where $\mathbf{b} = (b_1, \cdots, b_K)$, $\mathbf{n} = (n_1, \cdots, n_K)$, and $(\mathbf{b} \cdot \mathbf{n}) = \sum_{k=1}^{K} b_k n_k$. All classes require 0 loss probability so that we restrict our focus to multi-class stealing and do not address QoS differentiation with this model.

In the discussion below, we consider an example consisting of two traffic classes so that, for example, the transition from state $(0,1)$ to $(1,1)$ signifies admission of the first class 1 flow. The link capacity $C$ is 6 resource units and the flow bandwidths are 1 and 2, i.e., $b_1 = 1$ and $b_2 = 2$.

### 4.2 Markov Models

Below, we model the different schedulers and probing algorithms which we compare in Section 4.3.

**FIFO** Here, we consider FIFO as a baseline scenario. In our working example with two classes, each flow will probe the network and will be admitted only if $b_1 n_1 + b_2 n_2 \leq C$, including the probing flow. Figure 3 depicts the corresponding state transition diagram.

In the general case, the state space is $S = \{\mathbf{n} \in I^K : (\mathbf{b} \cdot \mathbf{n}) \leq C\}$ where $I$ is the set of non-negative integers and $I^K$ is the set of all $K$-tuples of non-negative integers. The link utilization is given by $u = \frac{1}{C} \sum_{n \in S} (\mathbf{b} \cdot \mathbf{n}) \pi(\mathbf{n})$ where $\pi(\mathbf{n})$ is the probability of being in state $\mathbf{n}$, which can be computed using standard techniques [10]. Notice that the probability of stealing is zero, since probing flows are only admitted if there is available bandwidth in the link.

**Flow-Based Fair Queueing** As described in Section 2, larger bandwidth flows can have bandwidth stolen in flow-based fair queueing systems. Figure 4 depicts the system's state transition diagram for our working example. As shown, the state space includes all states in which the number of flows multiplied by the lowest bandwidth flow is lower or equal to the link capacity. For example, a transition from state (4,1) to (5,1) is possible because 1 bandwidth unit is guaranteed for each flow, and this is sufficient for class 1 flows. Alternatively, a transition from state (5,0) to (5,1) is not possible because class 2 flows require 2 bandwidth units. Thus, the stealing states represent admissions of

[2] In general the rates of flows that belong to the same class may be different; however this assumption is required for the Markov chain formulation.

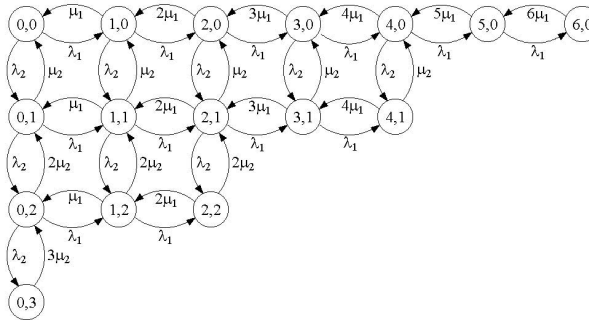



**Fig. 3.** FIFO Transition Diagram

low bandwidth flows when the system is at full capacity and high bandwidth flows are forced into a loss state. In the state transition diagrams, stealing states are represented by a crossed-out state.

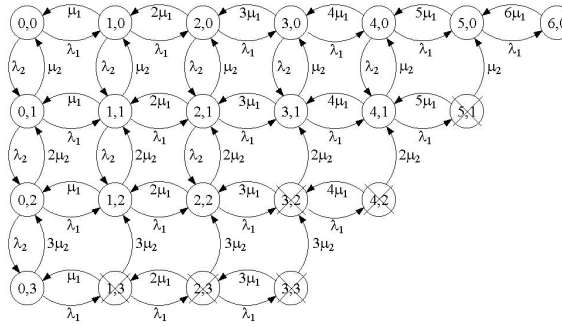



**Fig. 4.** Flow-Based Fair Queueing Transition Diagram

Suppose $\{n_1, n_2, ..., n_k, ..., n_K\}$ is the current admitted set of flows. Then a new class-$k$ flow is admissible if $\{b_1 n_1 + b_2 n_2 + ... + b_k([n_k + 1] + n_{k+1} + ... + n_K) \leq C\}$ with $\{b_1 < b_2 < ... < b_k < ... < b_K\}$. There are two cases: if the total demand satisfies $\{b_1 n_1 + b_2 n_2 + ... + b_k(n_k + 1) + b_{k+1} n_{k+1} + ... + b_K n_K \leq C\}$, then all flows are correctly admissible; however if $\{b_1 n_1 + b_2 n_2 + ... + b_k(n_k + 1) + b_{k+1} n_{k+1} + ... + b_K n_K > C\}$, then stealing occurs. Since the scheduler fairly allocates bandwidth to all flows, flows with bandwidth higher than the bandwidth of the flow requesting admission are forced into a loss state.

For the case of two types of flows, the state space including stealing states is given by

$$S_{FQ} = \{\mathbf{n} \in I^2 : ((b_1 n_1 + b_1 n_2 \leq C) \wedge (b_2 n_2 \leq C) \wedge (n_1 \neq 0)) \bigvee ((b_2 n_2 \leq C) \wedge (n_1 = 0))\}$$

For example, in our transition diagram, state (2,3) (with $n_1 = 2 \neq 0$) is a possible state (a stealing state) because $b_1 n_1 + b_1 n_2 = 5 \leq C$ and $b_2 n_2 = 6 \leq C$. State (2,4) is not a possible state since $b_1 n_1 + b_1 n_2 = 6 \leq C$ but $b_2 n_2 = 8 > C$. This example explains the need of the second inequality $(b_2 n_2 \leq C)$. With $n_1 = 0$, the state with the maximum number of flows is (0,3) because $b_2 n_2 = 6 \leq C$.

To generalize the state space to $K$ types of flows, let $k^*$ be the smallest $k$ such that $n_{k^*} \neq 0$, then

$$S_{FQ} = \{\mathbf{n} \in I^K : (b_{k^*} n_{k^*} + b_{k^*} n_{k^*+1} + ... + b_{k^*} n_K \leq C) \wedge (b_{k^*+1} n_{k^*+1} + ... + b_{k^*+1} n_K \leq C) \wedge ... \wedge (b_K n_K \leq C)\}$$

The mean utilization is then

$$u = \frac{1}{C} \sum_{\mathbf{n} \in S_{FQ}} (\mathbf{b} \circ \mathbf{n}) \pi(\mathbf{n}) \tag{3}$$

where

$$(\mathbf{b} \circ \mathbf{n}) = \begin{cases} (\mathbf{b} \cdot \mathbf{n}) & \text{if } (\mathbf{b} \cdot \mathbf{n}) \leq C \\ C & \text{otherwise} \end{cases} \tag{4}$$

The probability of stealing is

$$p_{st}^{FQ} = 1 - \frac{\sum_{\mathbf{n} \in S} (\mathbf{b} \cdot \mathbf{n}) \pi(\mathbf{n})}{\sum_{\mathbf{n} \in S_{FQ}} (\mathbf{b} \circ \mathbf{n}) \pi(\mathbf{n})} \tag{5}$$

computed as the percentage of bandwidth guaranteed to flows which is stolen by other flows.

**Rate Limiters** With rate limiters class $k$ flows are only allowed to use a maximum of $C_k$ bandwidth units. Here, we consider rate limiters of $C_1 = 2$ units and $C_2 = 4$ units respectively. Figure 5 depicts the corresponding state transition diagram. Given the functionality of the rate limiters, the state space is reduced to

$$S_{RL} = \{\mathbf{n} \in I^K : b_k n_k \leq C_k, k = 1, 2, ...K\}. \tag{6}$$

With this elimination of various high-utilization states, the overall system utilization in the general case of $K$ classes, given by $u = \frac{1}{C} \sum_{n \in S_{RL}} (\mathbf{b} \cdot \mathbf{n}) \pi(\mathbf{n})$, is then also reduced as compared to work-conserving systems. Clearly, the extent of this utilization reduction is a function of the system load, $b_k$ and $\lambda_k$, and $C_k$. If they are properly tuned, the penalty will be minimal, whereas if they become unbalanced due to load fluctuations, the system performance will suffer.

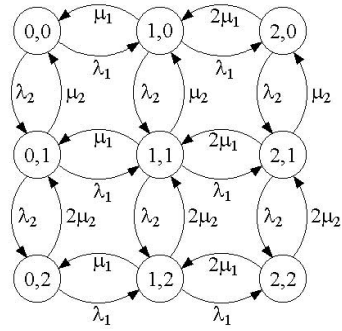



**Fig. 5.** Rate Limiters Transition Diagram

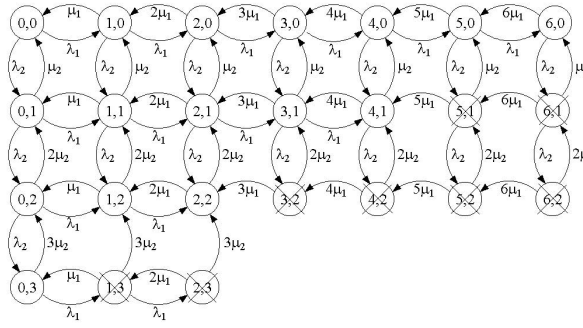



**Fig. 6.** Class-Based Fair Queueing Transition Diagram

**Class-Based Systems** In class-based fair queueing without rate limiters, resource borrowing across classes is allowed. However, as described in Section 2, stealing occurs as new flows in classes with reserved rate less than $\phi_i C$ request admission. Thus, with 1-level probing, a class $k$ flow with bandwidth $b_k$ will be admitted if, including the probing flow, one of two conditions occurs: $b_k n_k \leq C$ when $(\mathbf{b} \cdot \mathbf{n}) \leq C$, or $b_k n_k \leq C\phi_k$ when $(\mathbf{b} \cdot \mathbf{n}) > C$. In the state transition diagram of Figure 6, $\phi_1 = 1/3$ and $\phi_2 = 2/3$. As an example, consider the transition from (5,0) to (5,1). In the state space, the first set of inequalities is not satisfied because $b_2 n_2 = 2 \leq C$, but $b_1 n_1 + b_2 n_2 = 7 > C$. However the second set of inequalities is satisfied since $b_2 n_2 = 2 \leq C\phi_2$ and $b_1 n_1 + b_2 n_2 = 7 > C$. Therefore this transition is possible. Similarly, the transition from (4,1) to (5,1) is not allowed because neither set of inequalities are satisfied.

The state space has two parts. The first one is equal to the one of FIFO and allows borrowing between classes as long as $(\mathbf{b} \cdot \mathbf{n}) \leq C$. Thus $S_{CBQ-1}$ includes $\{\mathbf{n} \in I^K : (\mathbf{b} \cdot \mathbf{n}) \leq C\}$. Suppose we are in one of the edge FIFO states. Due to the borrowing between classes, for some classes, $b_k n_k \leq C\phi_k$, which we will call the underload classes ($UL$), and for others, $b_k n_k > C\phi_k$, which we will call the overload ones ($OL$). Suppose

there is currently no stealing in the system. New probing flows from $UL$ classes can be admitted until $b_{ul}n_{ul} = C\phi_{ul}$, with $ul \in UL$, irrespective of the value $(\mathbf{b} \cdot \mathbf{n})$. Thus departing from an edge FIFO state, new states can be created such that $\{\mathbf{n} \in I^K, ul \in UL : b_{ul}n_{ul} \leq C\phi_{ul}\}$. The overall state space $S_{CBQ-1}$ is then the union of the FIFO state space and the one constructed with these new states.

The utilization is $u = \frac{1}{C}\sum_{n \in S_{CBQ-1}} (\mathbf{b} \circ \mathbf{n})\pi(\mathbf{n})$ and the probability of stealing is $p_{st}^{CBQ-1} = 1 - \frac{\sum_{n \in S} (\mathbf{b} \cdot \mathbf{n})\pi(\mathbf{n})}{\sum_{n \in S_{CBQ-1}} (\mathbf{b} \circ \mathbf{n})\pi(\mathbf{n})}$. Note that $(\mathbf{b} \circ \mathbf{n})$ has the same definition as in flow-based fair queueing. Comparing the class-based and flow-based fair queueing, observe that the class-based system has a larger number of stealing states than the flow-based system. For example, transition from (6,0) to stealing state (6,1) is possible in the class-based system whereas state (6,1) does not exist in flow-based fair queueing. The reason for this is that the flow based fair queueing system blocks this 7th flow as it forces the system into loss. However, the class based system admits this flow since the requested rate of 2 bandwidth units is indeed available in class 2, even though it forces class 1 into a stealing situation. Regardless, even though there are more stealing states in the class-based system, the overall stealing probability is lower (as indicated by numerical examples and simulations below) because the fraction of time spent in such stealing states is lower in the class based system, so the bandwidth stolen will also become lower.

In contrast to the above 1-level probing, with $\varepsilon$-probing, all classes are probed to ensure that no stealing occurs. Here, the admissible states in this scheduler are the same as in FIFO, so the state space is the same, as well as the utilization. (We note that the utilization in the real system with nonzero probe durations is not the same however.)

### 4.3 Numerical Examples

Here we numerically solve the Markov models for each system described above. With the solution to the state probabilities, we compute the utilization and probability of stealing using the expressions derived above. We consider the scenario of previous sections with a link capacity of 6 bandwidth units. The weights of classes 1 and 2 are 1/3 and 2/3, respectively. The bandwidths of class-1 and class-2 flows are 1 unit and 2 units, respectively. Class 1's mean flow arrival rate is 8 requests per second while class 2's is 5. The mean life time of class 1 flows is 2/3 time units while class 2's is 1/4 time units.

**Table 1.** Utilization and Stealing Probability

| Probing Scheme | Utilization | Stealing |
|---|---|---|
| $\varepsilon$-**probing/FIFO** | 0.789 | 0 |
| **Flow-FQ** | 0.792 | 0.140 |
| **Rate Limiters** | 0.702 | 0 |
| **Class-FQ (1-level)** | 0.789 | $8.28 \cdot 10^{-4}$ |

We make two observations about numerical examples presented in Table 1. First, notice that $\varepsilon$-probing and rate limiters both have the effect of *eliminating* resource stealing.

However, $\varepsilon$-probing does so at higher utilizations. For example, $\varepsilon$-probing achieves 79% utilization as compared to 70% under rate limiters. Moreover, the difference between these two utilizations is determined by the relative class demands, which in this context are the relative flow arrival rates.

Second, note the stealing probabilities for flow- and class-based fair queueing (without $\varepsilon$-probing). Here, the stolen bandwidth is 0.140 for flow-based and $8.28 \cdot 10^{-4}$ for class-based. As evident from the model, CBQ incurs far less stealing than flow-based fair queueing. In simulation experiments, this relative difference still exists, however the probability of stealing for CBQ is far greater than it is in these numerical examples. The reason for this is even evident from the Markov model. In the CBQ system, stealing occurs as classes first demand bandwidths below and then later above $\phi_i C$ as defined in the state space of CBQ-1 level. It is precisely such system dynamics (changing resource demands) which are well captured by simulations but less via the Markov model. Thus, while the Markov model is useful to explore the origins and structure of multi-class resource stealing, we now turn to simulation experiments to quantitatively explore stealing under more realistic scenarios.

## 5 Experimental Studies

In this section, we present a set of simulation experiments with the goal of exploring the architectural design space as outlined in Section 2, evaluating $\varepsilon$-Probing presented in Section 3, and validating the conclusions of the analytical model of Section 4 in a more general setting.

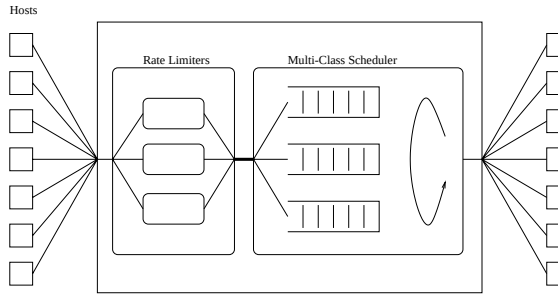
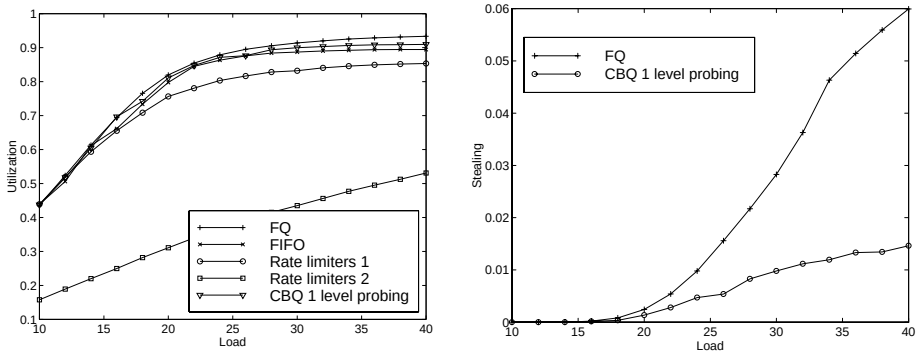



**Fig. 7.** Simulation Scenario

The basic scenario is illustrated in Figure 7. It consists of a large number of hosts interconnected via a 45 Mb/sec multi-class router. For some experiments, the router contains rate limiters which drop all of a class' packets exceeding the pre-specified rate. We consider several multi-class schedulers including CBQ, flow-based fair queueing, and rate limiters. We also consider FIFO for baseline comparisons. New flows arrive to the system with independent and exponential inter-arrival times through a Poisson process and probe for a constant time of 2 seconds. Flows send probes at their desired

admission rate except for $\varepsilon$-probes, which are transmitted at 64 kb/sec. New flows are admitted if the loss rate of the probes is below the class' threshold.




(a) Utilization (b) Probability of Stealing

**Fig. 8.** Utilization and Stealing vs. Load for Various Node Architectures

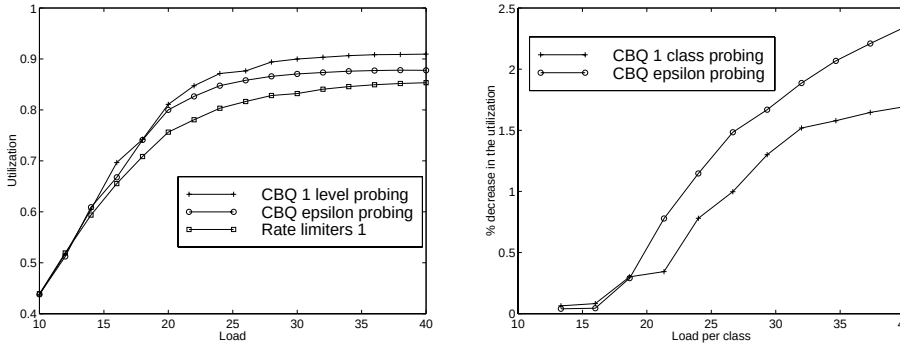
**Utilization and Stealing** In the first set of experiments depicted in Figure 8, we investigate the challenge of simultaneously achieving high utilization and a strong service model without stealing. In this scenario, there are three traffic classes with bandwidth requirements of 512 kb/sec, 1 Mb/sec, and 2 Mb/sec respectively. We consider three variants of the system depicted in Figure 7. The flow-based fair queueing curve, (labeled "FQ") represents the case in which the scheduler allocates bandwidth fairly among *flows*, i.e., the $N^{th}$ probing flow measures no loss if its rate is less than $C/N$. In contrast, the curves labeled "Rate Limiters 1", "Rate Limiters 2" and "CBQ 1 level probing" represent class-based scheduling. In the former case, each class is rate limited to $C/3$ so that all loss occurs in the rate limiters and none in the scheduler (cf. Figure 7). In the latter case, the classes are *not* rate limited and the scheduler performs CBQ with each class' weight set to 1/3. In all cases, probes are transmitted at the flow's desired rate and $\varepsilon$-probing of Section 3 is *not* performed. The x-axis, labeled load, refers to the resource demand given by $\frac{\lambda}{\mu}$.

We make the following observations about the figure. First, comparing the results with rate limiters and CBQ, Figure 8(a), indicates that CBQ achieves higher utilization than rate limiters due to the latter's non-work-conserving nature. That is, the rate limiters prevent flows from being admitted in a particular class whenever the class' total reserved rate is $C/3$, even if capacity is available in other classes. However, from Figure 8(b), it is clear that the higher utilization of CBQ is achieved at a significant cost: namely, CBQ incurs stealing in which up to 1.5% of the bandwidth (in the range shown) guaranteed to flows is stolen by flows in other classes. Hence the experiments illustrate that neither technique simultaneously achieves high resource utilization and a strong service model. Moreover, as the resources demanded by a class become mismatched with the pre-allocated weights, the performance penalty of rate limiters is further increased. That is, if the demanded bandwidth were temporarily 80/10/10 rather than 33/33/33, as is the

case for the curve labeled "Rate Limiters 2" at a load of 40, then the rate limiters would restrict the system utilization to at most 53% representing a 33/10/10 allocation.

Second, observe the effects of flow *aggregation* on system performance. In particular, flow-based fair queueing achieves higher utilization and has higher stealing than CBQ. With no aggregation and flow-based queueing, smaller bandwidth flows can always steal bandwidth from higher bandwidth flows resulting in both higher utilization since more flows are admitted (in particular low bandwidth flows) as well as more flows having bandwidth stolen. In contrast, with class based fair queueing, stealing only occurs when a *class* exceeds its 1/3 allocation (rather than a flow exceeding its $1/N$ allocation) and a flow from another class requests admission, an event that occurs with less frequency.

**$\varepsilon$-Probing** Figure 9(a) depicts utilization vs. load for three cases: CBQ with one-level probing, CBQ with $\varepsilon$-probing, and rate limiters. Observe that compared to one-level probing, $\varepsilon$-probing incurs a utilization penalty. There are two contributing factors. First, the $\varepsilon$-probes themselves cause an additional traffic load on the system despite their small bandwidth requirement. Second, by blocking flows which will result in stealing, there are fewer flows in the system on average with $\varepsilon$-probing than with one class probing. Regardless, this moderate reduction in utilization has the advantage of eliminating stealing completely. Moreover, the utilization penalty of *rate limiters* can be arbitrarily high depending on the mismatch between the demanded resources and the established limits. In contrast, the performance of $\varepsilon$-probing does not rely on proper tuning of rate limiters, but rather the overhead of $\varepsilon$-probing simply increases linearly with the number of classes.




(a) Utilization (b) Overhead

**Fig. 9.** Utilization and Overhead of $\varepsilon$-Probing

The utilization reduction solely due to probing is further illustrated in Figure 9(b). Observe that the overhead incurred in $\varepsilon$-probing is necessarily higher larger than that incurred by probing in only one class, as $\varepsilon$-probing must also ensure that other traffic classes are not in overload. However, due to the limited bandwidth required to probe in other classes, $\varepsilon$-probing incurs moderate utilization reductions typically below 2.5%.

Therefore, $\varepsilon$-probing is able to simultaneously eliminate stealing, provide multiple service levels, and enable full statistical sharing across classes.

## 6 Conclusions

Placing admission control functions at the network's endpoints has been proposed as a mechanism for achieving per-flow quality-of-service in a scalable way. However, if routers perform class differentiation such as multiple priority queues, the system becomes less observable to probing flows, precisely because of the performance isolation provided by the service discipline. In this paper, we have studied the resource stealing problem that arises in such multi-class networks and developed a simple probing scheme termed $\varepsilon$-probing which attains the high utilization of work-conserving systems while preventing stealing as in non-work-conserving systems with hard class-based rate limits. We introduced a Markov model that illustrated the design space of key network differentiation mechanisms, such as class- and flow-based weighted fair queueing and rate limiters. The model showed the different ways that stealing is manifested in the different configurations and provided a tool for formal comparison of diverse systems. Finally, our simulation experiments explored the design space under a broader set of scenarios. We quantified the severity of bandwidth stealing and found that $\varepsilon$-probing eliminates stealing with a modest utilization penalty required to observe the impact of a new flow on other traffic classes.

## References

1. G. Bianchi, A. Capone, and C. Petrioli. Throughput analysis of end-to-end measurement-based admission control in IP. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
2. S. Blake et al. An architecture for differentiated services, 1998. Internet RFC 2475.
3. L. Breslau, S. Jamin, and S. Shenker. Comments on the performance of measurement-based admission control algorithms. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
4. L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint admission control: Architectural issues and performance. In *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.
5. C. Cetinkaya and E. Knightly. Scalable services via egress admission control. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
6. V. Elek, G. Karlsson, and R. Ronngren. Admission control based on end-to-end measurements. In *Proceedings of IEEE INFOCOM 2000*, Tel Aviv, Israel, March 2000.
7. R. Gibbens and F. Kelly. Distributed connection acceptance control for a connectionless network. In *Proceedings of ITC '99*, Edinburgh, UK, June 1999.
8. E. Knightly and N. Shroff. Admission control for statistical QoS: Theory and practice. *IEEE Network*, 13(2):20–29, March 1999.
9. K. Nichols, V. Jacobson, and L. Zhang. Two-bit differentiated services architecture for the Internet, 1999. Internet RFC 2638.
10. K. Ross. *Introduction to Probability Models*. Academic Press, 1997.
11. I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *Proceedings of ACM SIGCOMM '98*, Vancouver, British Columbia, September 1998.

12. I. Stoica and H. Zhang. Providing guaranteed services without per flow management. In *Proceedings of ACM SIGCOMM '99*, Cambridge, MA, August 1999.
13. J. Wroclawski. Specification of the controlled-load network element service, 1997. Internet RFC 2211.
14. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 7(5):8–18, September 1993.
15. Z. Zhang, Z. Duan, L. Gao, and Y. Hou. Decoupling QoS control from core routers: a novel bandwidth broker architecture for scalable support of guaranteed services. In *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, August 2000.