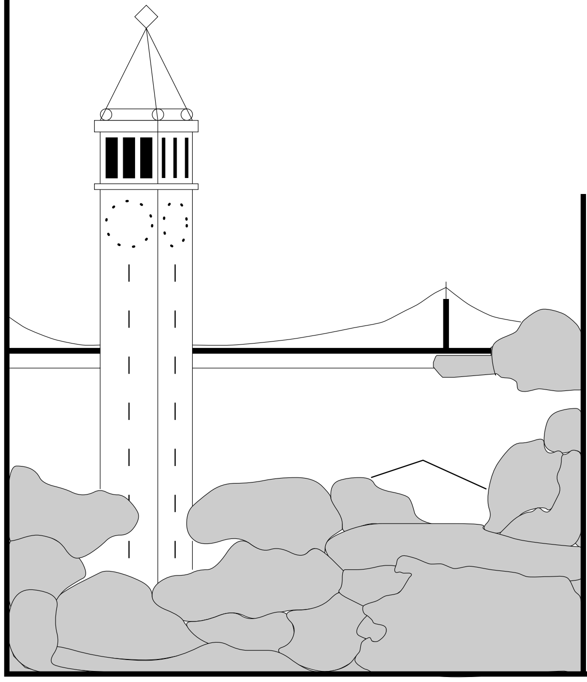
Furies: A Scalable Framework for Traffic Policing and Admission Control

Chen-Nee Chuah *Lakshminarayanan Subramanian* *Randy Katz*



Report No. UCB//CSD-1-1144

May 2001

Computer Science Division (EECS)
University of California
Berkeley, California 94720.

Abstract

Furies[1] provides a control framework for scalable, efficient admission control and traffic policing. Furies leverages the knowledge of traffic demand distributions between ingress-egress pairs and the network topology within an ISP in making admission control decisions. We propose to aggregate admitted flows for policing at edge routers instead of monitoring individual flows. Furies achieves this by assigning a unique flow-identifier to every admitted flow based on its ingress and egress point. As a result, the amount of states maintained by the edge routers can be reduced from $O(n)$ to $O(\sqrt{n})$, where n is the number of admitted flows, while core routers are stateless. Simulation results show that we can successfully detect a majority (64-83%) of the malicious flows with virtually zero false-alarms without maintaining per-flow state at the edge. Our implementation demonstrates that Furies adds minimal processing overhead to edge routers and can be incrementally deployed.

1 Introduction

There has been considerable research focused on extending the Internet architecture to provide better quality of service (QoS) for non-traditional Internet traffic such as real-time flows. Integrated Services (Int-Serv) with RSVP signaling [1] introduces end-to-end flow reservations, which requires core routers to maintain individual flow states and therefore does not scale well. Differentiated Services (Diff-Serv) [2], on the other hand, relies on packet marking and policing at the access or edge routers and different per-hop behaviors (PHB) at core routers to provide service differentiation. The packet forwarding mechanisms (e.g., scheduling, shaping, queue management, etc.) in Diff-Serv have been well-studied, but a fundamental understanding of the control framework is still lagging. Recent proposals use agents, known as bandwidth brokers (BB) [3, 4] to negotiate resource reservations between neighboring domains. However, it remains unclear how a BB interacts with the rest of Diff-Serv mechanisms, such as configuring the traffic policers or schedulers.

A control architecture for provisioning network resources [5] is an essential component for building a better service model for IP-based latency sensitive applications. Any such architecture must have two distinct components: *admission control* and *policing*. Admission control is necessary for limiting the usage of resources by competing flows, while policing is useful for detecting and penalizing *malicious* flows. Traffic policing in the Diff-Serv literature usually refers to parameter-based packet filter mechanisms, which are useful in tracking and shaping per-flow usage. In this paper, policing refers to monitoring an aggregate group of admitted flows and identifying malicious flows within this aggregate. We use the words "malicious" and "misbehaving" interchangeably to describe admitted flows that violate their allocated share of bandwidth.

Designing such an architecture faces numerous challenges. The most important one is to achieve compatibility with the existing Internet architecture. By compatibility, we mean that the proposed policies should be incrementally deployable and be able to reuse rather than replace the primitives supported by the existing network. Another important requirement is scalability. The overhead imposed by implementing the policies, such as amount of state maintained and processing time, should be minimized as the number of flows grows. To the best of our knowledge, scalable malicious flow detection has not been studied in the research literature. There is also a lack of an

[1]Furies is the name of the Roman goddess responsible for tormenting evildoers.

understanding how scalability, performance and deployment issues affect the design choices of such a control architecture. We attempt to bridge this gap in this paper.

This paper proposes Furies, a new framework for scalable traffic policing and admission control in the Internet. Recent BB proposals [4, 6] have suggested different policies to manage a domain's internal resources and allocate inter-domain resource agreements. Our work expands on these efforts and seeks insights on how to implement and integrate the required control mechanisms over the current Internet in an efficient and scalable way. Specifically, we present

- An admission control policy based on dynamic estimation of aggregate traffic demand between every pair of access routers in a network domain (typically a complete or subset of an ISP's network). Access routers (ARs) are ingress/egress points where traffic enters/exits a domain.
- A methodology for policing admitted flows and detecting malicious activity while maintaining a very small amount of state information at the access routers.

Furies uses the "core-stateless" principles [7], i.e., no state information is maintained in the core, and requires only simple priority scheduling in the core routers. Our architecture builds on many of the existing Diff-Serv primitives, like packet marking and leaky-bucket policing. The admission control and traffic policing mechanisms of Furies require passive monitoring of aggregate flows at the ARs.

This paper shows the scalability and the practicality of Furies through simulations, lab prototyping and implementation. In Section 2, we introduce the Furies architecture, and discuss how we arrive at several design choices based on our understanding of current Internet infrastructure. The key insight behind Furies is a coordinated way of assigning a flow-identifier, Fid, to every admitted *flow*, which allows aggregation of flows for traffic policing without compromising the ability to uniquely identify a flow if it is malicious. Each Fid has two subfields: $FidIn$ and $FidEg$. At ingress routers, admitted flows are aggregated based on their $FidIn$ for group policing. Similarly, egress routers police flows with the same $FidEg$ as an aggregate. The fact that each access router maintains only the aggregate state for each *group*, identified by Fid subfields, is crucial for the reduction of state from $O(n)$ to $O(\sqrt{n})$, where n is the number of admitted flows. The details of the algorithms of Furies are described in Section 3.

Furies does not strive to provide hard end-to-end guarantees but a relaxed nature of statistical QoS, as provided by soft real-time services. The simulations discussed in Section 5 are designed to evaluate the robustness and worst-case performance of our algorithms. We show that Furies can detect a majority of malicious flows with virtually zero false-alarms for a variety of source models. Section 6 provides a brief description of our implementation of Furies based on Click Router [8], and demonstrates that the overhead of deploying Furies at an access router is insignificant. If adopted, the Furies architecture could enable a dramatic shift in the paradigm of implementing real-time service models in the Internet. Deployment and remaining open issues are addressed in Section 7. We discuss the related work in Section 8, and present our conclusions in Section 9.

2 Design Rationale

In this section, we describe some basic properties of current Internet network topology and economic models based on our discussion with two major ISPs, and discuss how these considerations affect our design rationale.

A classic 1st-tier ISP backbone[2] generally consists of high-speed backbone links (0.6-10 Gbps) and low-bandwidth edge links (45-155 Mbps). Major ISPs in the United States typically have 15-25 Point-of-Presence (POPs) located in every big city in the country. The fan-out structure of POPs varies from city to city. For example, the number of access routers (ARs) connected to the core routers (CRs) inside a POP, usually in the range of 10-20 [9], depends on the number of customers in that region of the country. Different types of downstream "customers" are connected to the ISP at the ARs through the edge links. As mentioned in [10], the core backbone connects to neighboring private peers, public exchange points or transit providers via separate peering links. The access links, however, connect to corporate customers, university campuses, or web-hosting complexes, and 30-50 of such access links can be terminated at the same AR. Past studies have indicated that it is at these interconnection points, where large amounts of traffic converge and the backbone pipes meet the narrow access links, that congestion occurs, often resulting in packet loss and unreliable QoS.

Furies is designed to improve end-to-end performance by rationing the number of high priority flows admitted by the edge router based on continuous network measurements and estimated upper-bound traffic matrix. In our architecture, we treat the traffic demand coming from a private peer or transit provider differently from the high priority flows generated by end-hosts, because the resource allocation for both cases happen in vastly different time-scales and granularity. In the former, the traffic is usually subjected to peering agreements or Service-Level-Agreements (SLAs) [11] that reflect aggregate traffic performance (e.g., maximum round-trip delay). SLA renegotiation and the corresponding resource allocation decisions take place over longer time-scale, e.g., weeks or months. In the latter, the reservation requests from individual flows usually need fast admission control decisions, e.g., within ms, and the aggregate traffic demand fluctuate in a smaller time-scale, e.g., hours. We focus on the latter case in this paper. Although Furies also provides mechanisms for modeling, implementing and policing SLA traffic, a detailed description of them is out of the scope of this paper.

2.1 Features of Furies

The following characteristics are responsible for the scalability and robustness of our architecture.

2.1.1 Traffic Matrix Based Admission Control

In this paper, we argue that there is a need for a traffic matrix based admission control mechanism in the current Internet. A traffic matrix provides an abstraction of the traffic demand and the topology within an ISP. Furies uses an upper-bound traffic matrix (refer to Section 3.1) as a threshold to

[2]For proprietary reasons, we do not have access to the exact backbone topology. Example ISP network maps are available from `http://www.vbns.net/` and `http://www.cybergeography.org/atlas`.

perform admission control at the edge router without relying on an end-to-end signaling protocol such as RSVP. The upper-bound traffic matrix is topology sensitive and is derived based on passive measurements of the ingress-egress traffic demand within a domain. The available bandwidth on each link is split amongst the ingress-egress pairs in proportion to their estimated demands. This approach allows the admission control process to take into account the dynamic fluctuations of traffic demands, leading to more efficient resource provisioning. Such a traffic matrix based approach can also leverage intelligent traffic engineering mechanisms across different paths within the network.

2.1.2 Flows, Flow-Identifiers and Policing

In our framework, a *flow* refers to a high-priority stream between a specific source and a destination end-host that requests a particular amount of bandwidth. In order to pinpoint a misbehaving flow, Furies assigns each admitted flow a unique 32-bit flow-identifier, Fid, which is inserted in the packet header. This Fid is explicitly assigned by Furies and is not assumed as a random number by the flow. Every Fid consists of two 16-bit sub-fields: $FidIn$ and $FidEg$. All flows that enter or exit at an AR are aggregated into different groups based on their $Fids$. Each of these groups is identified with a unique group-identifier. The $FidIn$ and $FidEg$ subfields of a flow refer to the group identifiers used by its ingress and egress ARs, respectively.

Furies aggregates all flows that share the same subfield $FidIn$ (or $FidEg$) and polices them as a group at each ingress (or egress) AR. Every flow is policed at both its ingress and egress ARs in two distinct groups, thereby increasing the chances of detecting malicious flows. Every AR maintains only the aggregate state for each group and hence does not store any per-flow state.

2.1.3 Explicitly Assigned vs User-Selected Fids

In our approach, we attempt to maximize the level of flow aggregation without compromising the uniqueness of $Fids$, thereby minimizing the number of groups to be policed at every AR. An explicit assignment of $Fids$ can achieve this since it can keep track of the availability of individual $Fids$ and allocate unused ones to new flow requests. For example, if there are n flows from an ingress to a specific egress router, an explicit assignment can preserve uniqueness by maintaining only $\sqrt{n}$ groups at each of the two routers. It also allows aggregating flows with similar bandwidth requirements into a common group for policing.

On the other hand, if flows were allowed to assume their own $Fids$, then it would be necessary to maintain per-flow state information at the edge routers for traffic policing. Any scheme that attempts to group flows with random identifiers into common groups without maintaining per-flow state information must build a group membership function on the random $Fids$. The cost of performing an online grouping of flows based on these functions would be very high because the $Fids$ are continuously changing. Techniques like Stochastic Fair Blue (SFB) [12] that use random hash functions cannot be applied because they do not provide an inverse mapping from group identifiers to actual $Fids$. Thus, using SFB alone does not provide a direct mechanism to verify whether a suspected flow is truly misbehaving without knowing its actual Fid.

In other words, if a flow is detected as possibly misbehaving, we do not have a direct mechanism for verifying that without knowing its actual Fid.

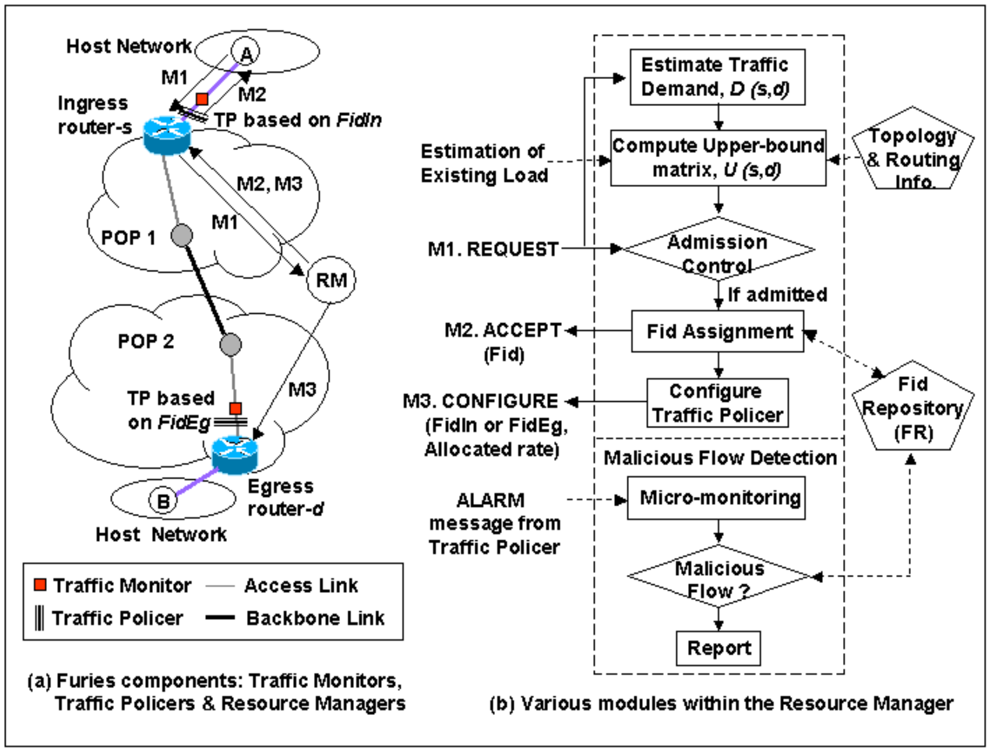


Figure 1: **Components of Furies.**

2.2 Components of Furies

One of the design requirements of Furies is to extend rather than replace the existing network architecture to support incremental deployment. Furies adds functionality to access routers and monitoring components of a POP in order to implement its policies. Furies has three components: a Resource Manager (RM), Traffic Monitors (TMs) and Traffic Policers (TPs). The RM, the most crucial component, interacts with the ARs continuously to admit new flows and identify misbehaving flows. Fig. 1(a) illustrates the approximate placements of these components in two POPs of an example ISP network. The arrows labeled M1, M2 and M3 show the REQUEST, ACCEPT and CONFIGURE control messages between the RM, ARs and customer routers in the host networks. Fig. 1(b) shows the various modules of the Resource Manager (RM) and the logical flow of the control messages.

The Resource Manager is a logical unit within a network (ISP) that can be physically placed at the fault monitoring point or policy server in an ISP. In practice, it could be implemented as a distributed architecture across POPs. The RM maintains the repository of assigned *Fids* and their allocated bandwidths in the Fid-Repository (FR). It also keeps track of the traffic demands between every pair of ingress-egress ARs and estimates the upper-bound traffic matrix for admission control. When a new REQUEST message arrives (M1 in Fig. 1a), the RM performs admission control and assigns an *Fid* if admitted. Upon successful admission, the RM then sends an ACCEPT message (M2) along with the *Fid* back to the host. Otherwise, the RM sends a REJECT message. It also updates the traffic policers at the ingress and egress ARs when a new flow is admitted using the

CONFIGURE message (M3 in Fig. 1a).

A Traffic Monitor (TM) at each ingress AR passively measures the rate of admitted traffic between itself and every other egress AR. It updates the RM periodically, and these updates are used by RM to estimate the load on each link. A Traffic Policer (TP) is introduced at each ingress and egress AR to police groups of flows identified by subfields of their *Fids* (Section 3.2.1) and identify the group that violates the aggregate allocated bandwidth. The TP micro-monitors the flows in a misbehaving group and reports potential malicious flows (flows with high usage) to the RM.

2.3 Assumptions

We make the following assumptions in our design:

1. We focus on access routers, but not hosts, as endpoints. We assume all-pair shortest paths between a specific ingress and egress router can be determined by the underlying intra-domain routing and BGP route-selection process, after tie-breaking. We do not modify any routing decisions.

2. Non-conformant packets that violate traffic profiles can either be dropped or re-marked to lower priority levels at the TPs. Our architecture can support both cases equally well, but for simplicity, we chose packet dropping as the default choice.

3. We assume we can insert a 32-bit *Fid* in the packet header. In practice, this is done by the customer router.

4. Furies requires explicit REQUEST and TEARDOWN messages for admitting and releasing every flow. The messages can be generated by either the customer router or a proxy and sent as UDP packets at the same level of priority (high) as the data packets.

3 Detail Descriptions of Furies

In this section, we provide a detailed description of our algorithms for admission control and malicious flow detection.

3.1 Traffic Matrix Based Admission Control (TMAC)

Since the admission-controlled traffic must coexist with current best-effort traffic, we follow the same set of principles outlined in [14]. First, we need to strictly limit bandwidth allocated to the admission-controlled traffic so that it never borrows bandwidth from the best-effort class. Secondly, best-effort traffic must not pre-empt admission-controlled traffic, and hence the latter should be served in a higher priority class. In Furies, the maximum bandwidth usage along a link by high-priority traffic is bounded by a small fraction, denoted as β_{T}, of its total capacity. This ensures that the edge-to-edge queuing delay within a domain is statistically bounded.

For ease of discussion, we define the following terms:

- **IE-Pipe(s, d)** refers to all high-priority traffic entering the ISP at ingress AR-s, and exiting at egress AR-d.
- **Traffic Matrix**, D, captures the distribution of traffic demand within an ISP. Given an ISP network with K access routers, D is an $K \times K$ matrix, where each entry $D(s,d)$ represents the total bandwidth requirement of high priority flows from ingress AR s to egress AR d.
- **Upper-bound Matrix**, U, is computed by splitting the bandwidth of the bottleneck links from AR-s to AR-d between all the competing IE-Pipes(s,d) in proportion to their estimated traffic demands $D(s,d)$ (Section 3.1.2). $U(s,d)$ is used as a threshold for admission control.

When a new flow arrives at AR-s, the AR-s forwards the REQUEST message to the RM. The REQUEST message indicates the average rate r and burst parameter b required by the flow. The peak rate, r_{peak}, is computed as $r + \frac{b}{\delta}$ where δ is the samping period (0.5 s in this paper). The input link and destination IP address of the REQUEST message are used to identify the end-points of demands, i.e., ingress AR-s and egress AR-d. This process requires information about destination prefixes associated with each egress link. We follow the same methodology outlined in [10], and keep a table of destination prefixes and the corresponding egress routers in the RM. Each `dest-prefix` consists of an aggregated network address advertised by the egress router and a mask length. The `dest-prefix` allows a set of destination IP addresses to be mapped to a specific egress router d. In practice, these `dest-prefix`'s can be determined from the forwarding tables of routers that terminate egress links. We assume RM can obtain this information from the underlying routing protocol.

Upon receiving a new REQUEST message for a flow with peak rate r_{peak} between ingress AR-s and egress AR-d, the RM checks the following condition:

$$M(s,d) + r_{\text{peak}} < (1+\sigma) \cdot U(s,d) \qquad (1)$$

where $M(s,d)$ is the estimated rate of admitted traffic between AR-s and AR-d, $U(s,d)$ is the admission threshold, and $0 \le \sigma \le 1$ is the hysteresis parameter. The new flow is admitted if the condition in (1) is not violated. Otherwise the flow is rejected. Setting $\sigma > 0$ exploits statistical multiplexing of multiple flows and increases utilization level, but σ must be small enough to avoid oversubscription of resources and its corresponding adverse impact on the end-to-end performance. We found that $\sigma \in [0.10.2]$ is a reasonable range. We describe how D, M and U matrices are estimated in the next two subsections.

3.1.1 Estimating Traffic Matrix and Usage Matrix

The RM estimates the traffic demand, $D(s,d)$ as simply the sum of the peak rate requested by individual flows (including both admitted and rejected requests):

$$D(s,d) = \sum_{\{f\}} r_{\text{peak}}(f) \qquad (2)$$

where $\{f\}$ is the set of flows that enter the network at ingress AR-s and exits at egress AR-d.

We use a time-window estimator as described in [15] at each ingress AR-s to estimate the usage vector $M(s,:)$, where each entry $M(s,d)$ represents the rate of admitted traffic to each egress AR-d.

The measurement window Δ_{T} is a multiple of Δ_{S}, and at the end of every Δ_{T}, $M(s,d)$ is set to the highest average load computed for any Δ_{S} in the previous window. All ingress AR-s send regular updates of their $M(s,:)$ vectors to the RM, which maintains the complete matrix M.

3.1.2 Computing Upper-bound Traffic Matrix

We assume that shortest paths for all ingress-egress pairs can be determined from the underlying routing protocol. Let $\mathcal{P}_{ij}$ be the set of links that form the shortest path from ingress router i to egress router j:

$$\mathcal{P}_{ij} = \{l_1, l_2, \ldots, l_h\}$$

where h is the number of hops from i to j.

For each link with capacity C_l, we limit the share of bandwidth allocated to high priority traffic as $\beta_{\mathrm{T}} \cdot C_l$ where $0 < \beta_{\mathrm{T}} < 1$. This available bandwidth should be split among different IE-Pipes that share the same link in proportion to the corresponding $D(s,d)$. For a particular IE-Pipe(s,d), the allocated bandwidth on link l is:

$$U_l(s,d) = \frac{D_l(s,d)}{\sum_{i,j \text{ s.t. } l \in \mathcal{P}_{ij},\, i \neq j} D_l(i,j)} \cdot \beta_{\mathrm{T}} \cdot C_l \tag{3}$$

The admission threshold for IE-Pipe(s,d) can be determined as:

$$U(s,d) = \min_{l \in \mathcal{P}_{sd}} U_l(s,d), \ \ \forall s, d, s \neq d. \tag{4}$$

We compute the entries, $U(s,d)$, from Eqs. (3) and (4) when $s \neq d$ and set $U(s,d) = 0$ when $s = d$. Since traffic demands for operational IP networks exhibit different time-of-day patterns (as reported in [10]), we need to update U often enough to reflect the dynamic fluctuations. The update interval is denoted as t_{u}, which is a parameter that we vary in our experiments.

Robustness to non-stationarity in the demand can be achieved by choosing the update interval, t_{u}, to be smaller than the time-scale at which $D(s,d)$ fluctuates. The authors in [10] observed a small number of "heavy-hitters" (i.e. large traffic demands) that exhibit certain amount of stability across time (different days, different weeks). Setting t_{u} in the order of minutes should be sufficient.

3.2 Traffic Policing

The policing mechanism of Furies is mainly designed for detecting a small number of malicious flows among a large group of flows. Furies aggregates admitted flows into different groups at every AR and every flow is policed in distinct groups at both its ingress and egress ARs. When an AR detects an aggregate group of flows as misbehaving, it micro-monitors this group to identify possible individual flows that may be misbehaving. Normally the candidate flows in this group are the ones that transmit at a very high rate.

3.2.1 Flow-Identifier Assignment and Releasing

Furies assigns each access router i a set of M unique group identifiers, denoted as $\mathcal{A}_i = \{x_{i1}, x_{i2}, \ldots, x_{iM}\}$, where each member is an 16-bit binary number and is unique across the set $\mathcal{A}_i$. The sets $\mathcal{A}_i$ and $\mathcal{A}_j$ associated with any two ARs i and j are mutually disjoint.

As mentioned earlier, every flow is associated with an Fid which has two fields: $FidIn$ and $FidEg$. Any Fid of an admitted flow should satisfy the following properties:

1. If the flow is routed from AR-s to AR-d, then $FidIn \in \mathcal{A}_s$, and $FidEg \in \mathcal{A}_d$.
2. No other flow should have the same Fid.

When a new flow from AR-s to AR-d is admitted, the RM picks a random x_{sk} from $\mathcal{A}_s$ and a random x_{dl} from $\mathcal{A}_d$ such that the Fid with $FidIn = x_{sk}$ and $FidEg = x_{dl}$ is not used by any other flow. This Fid is assigned to the flow, and a new entry with this Fid and its allocated bandwidth is added to the Fid-Repository (FR). The total number of flows that can be uniquely identified in this scheme is $K \cdot M^2$ for a particular ingress router, where K is the total number of potential egress routers, each has its own unique set of identifiers.

As a reality check, we consulted two major ISPs to get an idea of the typical number of flows on an ingress link: roughly 300-5000. Based on the discussion in Section 2, K, which is roughly (number of POPs $\times$ number of access routers/pop), is in the order of 150-500. We need a maximum of $M = \sqrt{5000}$, which is roughly 71 unique identifiers per $\mathcal{A}$ set. Since the total number of unique identifiers required for the whole ISP is $M \times K$, allocating 16-bits for each subfield should be more than sufficient for producing mutually disjoint $\mathcal{A}_i$ for all AR-i.

We assume the admitted flow will send a TEARDOWN message to the ingress router when it terminates. The TEARDOWN message contains the Fid, and its allocated peak rate r_{peak}. When RM receives the TEARDOWN message, it updates the FR accordingly and releases the Fid. Furies does not explicitly update the estimated load $M(s,d)$ but relies on the time-window estimator to detect the departure of the flow.

3.2.2 Malicious Flow Detection (MFD)

Furies deploys a set of token bucket traffic policers [15] at each ingress and egress AR to police packets from the admitted flows with the matching $FidIn$ or $FidEg$ group identifiers. Every group identifier, $x \in \mathcal{A}_i$, is associated with a token bucket with two parameters, r_{tot} and b_{tot}, where r_{tot} is the total average rate of admitted flows and b_{tot} is the total burst size. When a new flow between AR-s and AR-d is admitted, the RM sends a CONFIGURE message that specifies the $FidIn$ and $FidEg$ of the admitted flow to the ingress AR-s and egress AR-d, respectively, along with its average rate r and burst-size b. TP$_s$ and TP$_d$ will then update the the token bucket with the matching $FidIn$ and $FidEg$ accordingly. The packets that violate the associated traffic profile are considered *non-conformant* and discarded. Each TP keeps a counter for the non-conformant packets and reports the statistics to RM.

As an example, let a flow with $Fid = [f, g]$ be malicious. All flows with the same $FidIn = f$ will be policed as an aggregate in the same token bucket at the ingress point, TP$_s$, regardless of what their $FidEg$ is. If a group token bucket with $FidIn = f$ is violated, the affected TP reports f to the RM using an ALARM message (Fig. 1b). However, this information alone is insufficient for pinpointing the exact misbehaving flow, because there can be as many as M flows with the same $FidIn$, and any of these could potentially contribute to the violation of the total allocated rate. If the TP at an egress router $FidEg = g$ also sends an ALARM message, the RM can infer that a flow with Fid that contains both f and g as its subfield is malicious. However, this may not always happen. To improve the effectiveness of MFD, Furies requires the TPs to perform "micro-policing" for a duration of t_{mp} whenever a group token bucket is violated. During this period, we sample the peak rate of each flow and identify the top n_{mp} potentially malicious flows. For each of these flows, we report its Fid (from the packet header) and its sampled peak rate to the RM. We consider n_{mp} = 5 in this paper.

For each reported flow with $Fid = b$, the RM compares the allocated rate, R_b with the measured peak rate r_{b} reported in the ALARM message:

$$r_{\mathrm{b}} < (1+\epsilon) \cdot R_b \tag{5}$$

where ϵ is a hysteresis parameter to absorb transient behavior of bursty traffic. If the condition in (5) is violated, the flow is considered misbehaving. ϵ is typically between 0 and 0.05. A counter associated with this flow is incremented for every such violation of condition (5). To reduce the probability of false alarm, we introduce a second hysteresis parameter, η, which determines the minimum number of violations before a flow is reported as misbehaving. A reasonable range for η is between one and five.

Several actions could possibly be taken against the misbehaving flow, e.g., dropping all the future packets of this flow, demoting all its packets to best-effort, or charging more for the connection. However, the study of these penalty actions is out of scope of this paper. Our goal is to provide insight into the scalability and robustness issues of the MFD-scheme itself.

Since the policing at TP is performed on a group of flows sharing the same 16-bit subfield of Fid, the amount of state information maintained at the ingress router is proportional to M, i.e., the number of unique identifiers in the set, $\mathcal{A}$. In practice, the maximum value of M is 71, which is significantly smaller than typical number of active flows at the AR, which is roughly 5000.

3.2.3 Hiding in the Aggregate

Although group policing allows the Furies architecture to scale, it limits the effectiveness of MFD to detect a misbehaving flow that "hides" in the aggregate. There are two specific cases where a malicious flow can hide in the aggregate:

1. The total usage of the group is less than the tolerated usage because either certain flows are under-utilizing their resources or the percentage of over-utilization is less than the threshold ϵ.

2. The malicious flow is a relatively small flow in a misbehaving group and is not reported since there are many other larger flows in the group.

To address this problem, Furies introduces redundancy by deploying TP at each egress router that terminate a set of egress links. By assigning a unique Fid to every flow, we ensure that no two flows are in the same group in both the ingress and the egress routers. Essentially every flow is policed in the aggregate at two distinct points to maximize the number of malicious flows that are detected.

3.2.4 Other Issues

Bogus Identifiers
It is possible for an external malicious user/application to create its own Fid which is valid but has not been explicitly assigned by the RM. We refer to such identifiers as *bogus* flow identifiers. This problem can be easily solved by using a secure hash function with a secret key within an ISP. Assume that the RM and the ARs share a secret key, K, for a hash function $h()$. Given a $Fid = f$,the actual Fid in the packet of the flow is set to $h_K(f)$ and the edge routers needs to perform an inverse hashing of the Fid for every packet before performing the policing operations.

Routing Changes
When a routing change occurs, the previous Fid will not match with the group identifiers in at least one of its new ARs. Whenever the subfield of the Fid of a flow does not match any of the group identifiers at an AR, we can either (1) remark the packets of this flow as best-effort, or (2) contact the RM for re-admission of this flow with the new endpoints. Further investigation is needed to understand the the performance and security issues of both approaches.

4 Simulation Framework

The aim of the simulation study is to evaluate the performance and robustness of Furies design. We use the ns-network simulator to implement the basic mechanisms of Furies. The TP[3] is implemented as a connector in front of a node, and a time-window estimator is introduced at each input link to estimate the rate of existing flows. The admission control module is created as an NsObject and inserted before the ingress router. The various tasks of the RM in Furies are implemented at the Tcl-level. Our Furies-patch works for ns-2.1b6.

4.1 Network Topology

Since it is infeasible to run large-scale Internet simulations over actual networks, we simulate a simple subgraph of the backbone topology shown in Fig. 2 that consists of 6 POPs. For simplicity, we analyze only one access router per POP. AR0-AR2 are ingress routers where traffic enters the backbone, while AR3-AR5 are egress routers where traffic leaves the backbone to other POPs. Traffic demand originating from AR-s to AR-d is denoted as $D(s, d)$. The bandwidth of the bottleneck link in our simulations is 10 Mbps. The target utilization level β_{T} for high priority traffic is 0.3.

[3]We modify the DiffServ module contributed by Sean Murphy, `http://www.teltec.dcu.ie/~murphys/ns-work/diffserv/index.html`.

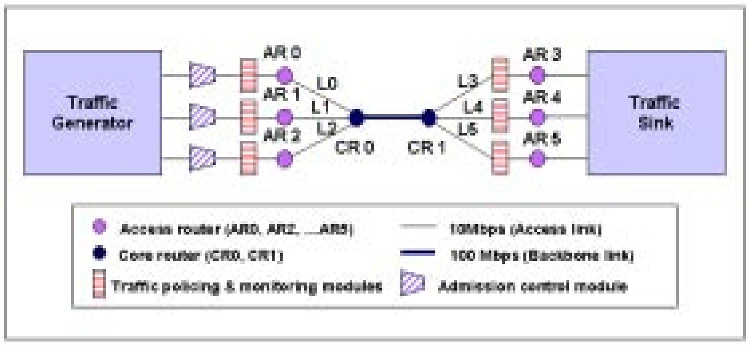


Figure 2: Simulation Topology

Each router uses a simple priority scheduler, and the control messages are sent at the same priority as the data. There is enough buffering for 200 packets at each queue.

4.2 Traffic Generation

For proprietary reasons, we are unable to access the real traffic measurements from Internet backbone networks. We derive traffic models based on published results and discussions with researchers who have studied data sets collected by ISPs themselves.

In our simulations, we model the admission-controlled traffic as a Poisson arrival process. The arrival rate from a host network to an ingress router is denoted as $\lambda_i(t)$. We use the indices t to indicate the time-of-day dependence of the traffic demand as reported in [10] and [16], but do not attempt to differentiate the effect from different user groups (e.g., domestic consumer vs. domestic business, etc.). Data sets from the Sprint IP backbone monitoring project [16] indicate that the bandwidth consumption peaks between 10 a.m. and 2 p.m. during the day and shows a dip from midnight to 3-4 a.m.

To reflect the realistic traffic demand, we introduce $\pm$ 10-15% changes to $\lambda_i(t)$ at a regular interval, T_ϕ. The traffic distributions from an ingress node s to a set of egress nodes are based on a probabilistic model where the probability associated with each egress node is proportional to the populations in the city where the egress POP is located.

We use four kinds of traffic source models in our experiments:

1. We use EXP1 to model a typical Voice-over-IP source. EXP1 has exponential on and off times with an average of 1.004 s and 1.587 s, respectively. This corresponds to a 38.53% talk-spurt cycle, as recommended by ITU-T specification for conversational speech [17]. The peak transmission rate is 64 kbps, and the average is 24.8 kbps.

2. EXP2 also has exponential on and off times, but with an average of 100 ms and 900 ms, respectively. The peak rate is increased to 248 kbps while keeping the average rate the same as EXP1, leading to a burstier source.

3. CBR is a constant bit rate source of 64kbps.

4. PARETO source has pareto on and off times but the average rate is the same as EXP1.

EXP1, EXP2 and CBR have exponential lifetimes with an average of 300s. The flow lifetimes of PARETO sources follow a log-normal distribution with average of 300 s. The aggregation of pareto sources is known to exhibit long range dependencies [18, 19]. For all four cases, packets are 320 bytes in length.

5 Performance Evaluation

In this section, we evaluate how well the admission control and traffic policing schemes in Furies perform with respect to the following goals:

- achieving high network utilization while maintaining low packet loss, and
- maximizing successful detection of misbehaving flows with as few false alarms as possible.

All simulations were repeated using different seeds to the random number generator. The averages across all repetitions are reported in our results. In all our experiments, we use homogeneous flows unless specified otherwise.

5.1 Basic Scenarios

Our first set of experiments simulate a basic scenario consisting of the topology shown in Fig. 2 with AR0-AR2 generating high priority flows to AR3-AR5. The basic scenario attempts to understand the effectiveness and practicality of the Traffic Matrix Admission Control (TMAC). We evaluate the utilization and loss characteristics for a variety of source models. We assume that all flows are well behaved. The offered load is chosen such that blocking rates in these experiments are approximately 20%. Utilization level is defined as measured link utilization/(β_{T} x link capacity).

5.1.1 TMAC Characteristics

In our first experiment, we examine how well TMAC reacts to dynamic fluctuations in traffic demand. Fig. 3 shows the bandwidth-sharing achieved on the bottleneck link L3 (Fig 2) as a result of the TMAC policy when EXP1 source model is used. The x-axis shows simulation time and the y-axis shows the average bandwidth used by each of the three IE-Pipes: (0,3), (1,3) and (2,3) over ten-second intervals. All the EXP1 sources from AR0-AR2 which are destined to AR3 compete for bandwidth on L3. The upper-bound traffic matrix is updated at regular intervals of t_{u}=2 minutes in this case. We artificially introduce abrupt changes in traffic demand $D(0,3)$, $D(1,3)$ and $D(2,3)$ at time 900 s, 1800 s, and 2700 s. The dashed lines on Fig. 3 show the ideal allocation of bandwidth among these three ingress-egress pairs if the traffic demand is known a priori. The solid lines show the actual link sharing achieved by Furies. Results indicate that each ingress-egress pair gets roughly its ideal bandwidth allocation, with 8.1% - 19.9% deviation from the ideal values. On an average, it takes 2.7 minutes after each perturbation to converge to the ideal utilization level.

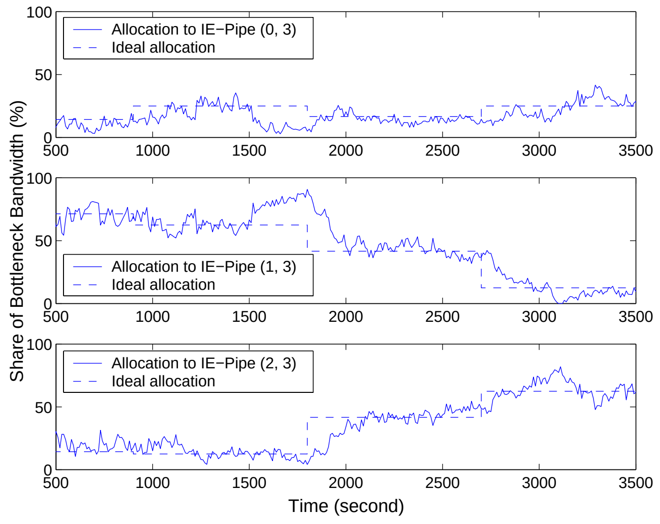


Figure 3: **Basic Scenario:** Share of bottleneck bandwidth allocated to each ingress-egress pair that share Link-3: $D(0,3)$, $D(1,3)$ and $D(2,3)$, $t_u = 2$ minutes.

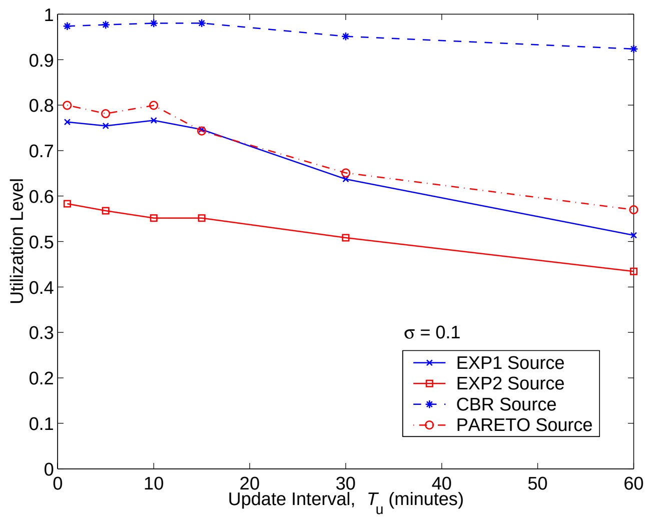


Figure 4: **Basic Scenario:** Sensitivity analysis of link utilization with respect to the update interval for upper-bound traffic matrix, t_u and source models.

Since the admission control policy in Furies depends on the estimation of the upper-bound traffic matrix U, we need to understand how the utilization level is affected by the frequency at which U is updated. Fig. 4 shows the network utilization level averaged over three bottleneck links as t_u is varied from 1 to 60 minutes for all four different source models (EXP1, EXP2, CBR and PARETO). We inject ± 10-15% random fluctuations in traffic demand as described in Section 4.2 at regular intervals of T_ϕ=30 minutes. For a given source model, each point on the curve shows the utilization level for a particular t_u averaged over 10 simulation runs with different random seeds. In all four cases, the average utilization level decreases as t_u is increased because the estimated U is less responsive to traffic fluctuations. With EXP1 sources, the utilization level decreases from 77%

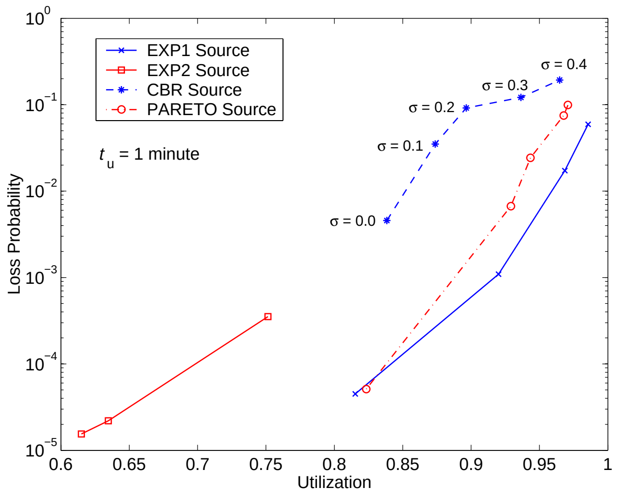


Figure 5: **Basic Scenario:** Loss-load curves for four different traffic models as the hysteresis parameter σ is varied, t_{u}=2 minutes.

to 51%. Clearly, experimenting with CBR sources achieves the highest utilization level (92-98%) since the bandwidth usage of admitted flows is fairly deterministic. The presence of burstier source (EXP2) reduces the utilization level (43-58%), while the long range dependent traffic (PARETO) achieves roughly the same utilization as (EXP1). In all cases except CBR source, packet loss is less than 0.3%. For CBR source, packet loss varies from 0.6% to 4.5%.

5.1.2 Trade-offs Between Loss and Utilization

There is a trade-off between packet loss and network utilization level in any admission control system. To increase the network utilization, one might admit more flows into the system. But this may lead to over-subscribing resources and a higher loss rate. Tuning the hysteresis parameter, σ, allows an ISP to operate at the desired utilization level. We are interested in the range of utilization and loss rates that can be achieved by varying σ for different source models. Results are plotted as a set of *loss-load* curves in Fig. 5. Each point shows the average packet loss for a given utilization level as σ is varied from 0.0 to 0.5. The loss load curves have the same frontier as those of the measurement-based admission control algorithm (not shown here) reported in [15]. The burstier source (EXP2) shows a dramatically different range as compared to the three other traffic sources. Since the peak rate of EXP2 is 10 times larger than that of EXP1, and the admission control policy is based on peak rate, we tend to admit less number of EXP2 flows. However, since the activity cycle of EXP2 is very short (10%) and the used bandwidth is zero for a large amount of time, we observe this under-utilization of network resources.

5.2 Detection of Malicious Flows

This section investigates the effectiveness of the malicious flow detection (MFD) scheme in Furies. We are interested in the following two events:

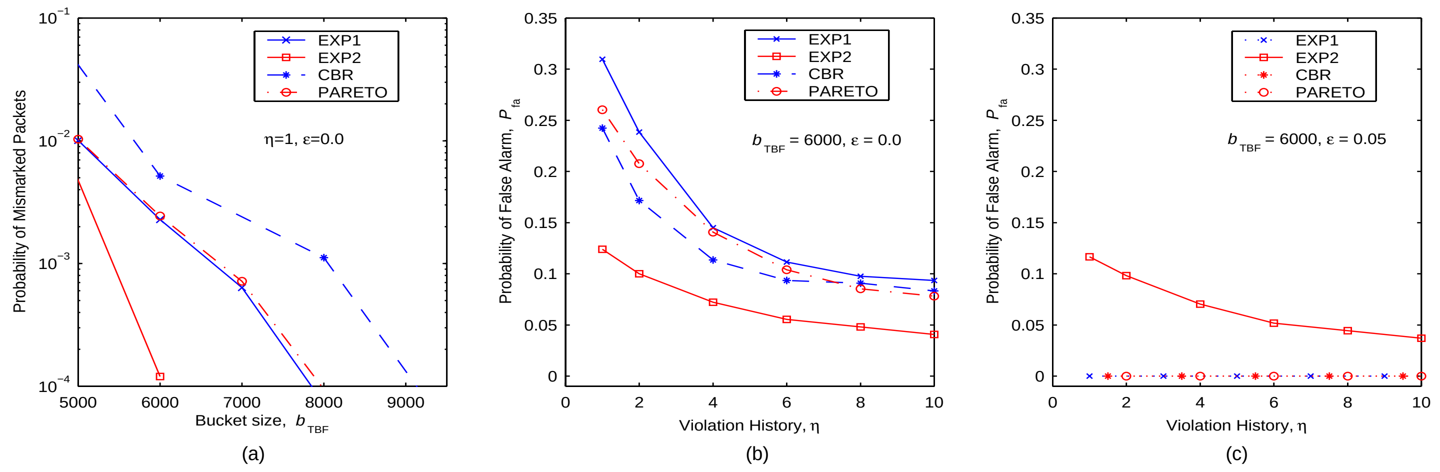


Figure 6: **Case 1:** Zero Malicious Flows.

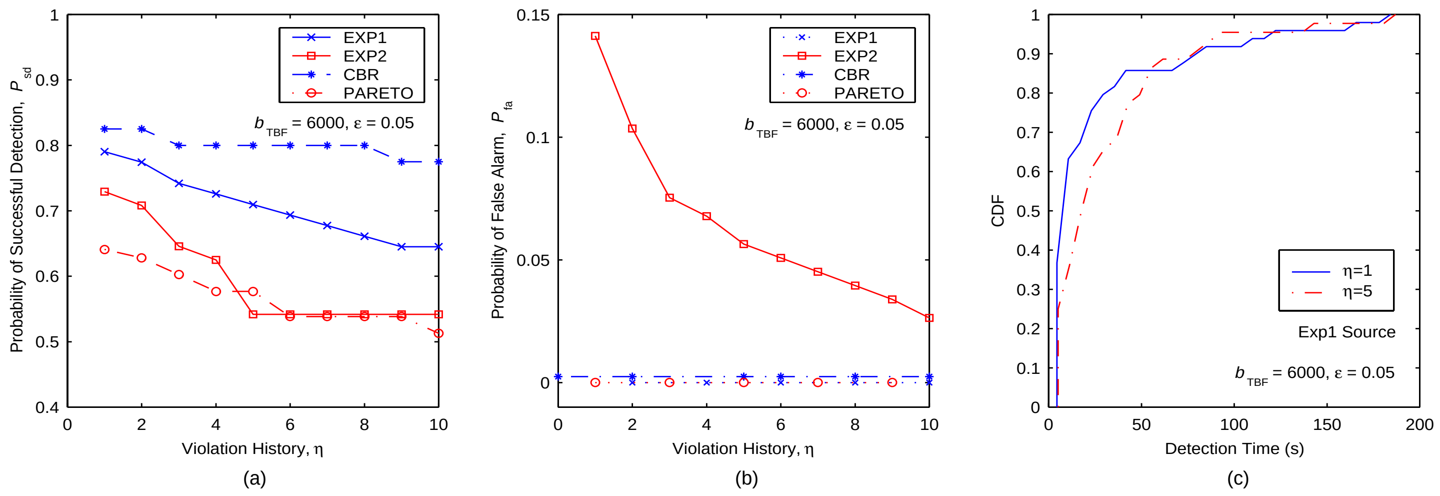


Figure 7: **Case 2:** Many small homogeneous flows; a small fraction, γ =0.1, misbehave.

- **Successful Detection:** a misbehaving flow is correctly identified, and
- **False Alarm:** a well-behaved flow is misclassified as malicious.

We define the probability of successful detection P_{sd} as the fraction of malicious flows that are actually detected. Similarly, the probability of false alarm P_{fa} is the fraction of normal flows that are incorrectly reported as misbehaving. Since the flows are policed as an aggregate, malicious flows can cause packets from complying flows to be incorrectly marked as non-conformant. The probability of such incorrectly-marked packets is denoted as P_{mis}.

To examine the robustness of the MFD scheme, we simulate four extreme cases:

Case 1: We consider a basic scenario as described in 5.1 with zero malicious flows.

Case 2: This load model is similar to Case 1, but now a small fraction, γ, of the flows are misbehaving.

Case 3: We consider a mixture of one large flow and many simultaneous small flows. The peak rate of the large flow is 10 times larger than the peak rate of a small flow. All of the small flows are compliant, and only the large flow misbehaves.

Case 4: Again, we consider a mixture of one large flow and many simultaneous flows like Case 3. However, the large flow is compliant this time, and a fraction γ of the small flows are malicious.

We consider homogeneous flows in each case, and repeat each experiment using four different source models: EXP1, EXP2, CBR and PARETO. For each scenario, the simulation was repeated 10 times with different random seeds, and the average P_{sd}, P_{fa}, and P_{mis} was computed. Each simulation ran for 1000s. All experiments were performed under high load, and the hysteresis parameter for admission control, $\sigma = 0$. A *malicious* source requests allocation for r kbps but sends traffic at a higher rate, randomly chosen between $1.1 \cdot r$ and $1.2 \cdot r$ kbps (10-20% violation). The average and peak rate for each source model is the same as described in Section 4.

The experiments in Case 1 are intended for understanding the limitations of the Furies-MFD and its performance sensitivity with respect to different choice of design parameters. Ideally, none of the flows should be reported as "misbehaving", but the transient behavior of bursty traffic could momentarily overflow the token bucket filters (TBF), and be interpreted as malicious, leading to a "false alarm". Fig. 6a shows how the choice of bucket size, b_{TBF} at the Traffic Policers (TP) affects the probability of mis-marked packets, P_{mis}. A smaller value of b_{TBF} is more effective in policing the aggregate traffic and detecting misbehaving flows, but there should be enough tokens to allow the legitimate packets to pass, and keep the P_{mis} low. Except for CBR traffic, P_{mis} is below 0.01 for other source models. For CBR source, increasing b_{TBF} to 6000 is sufficient to reduce P_{mis} to 0.005. The two hysteresis parameters η and ϵ determine under what conditions a flow is reported as "malicious" (Section 3.2.2), but have no effect on P_{mis}.

We can relax the condition for MFD by increasing ϵ and η, and this helps to reduce the number of false alarms. Fig. 6b and 6c study how P_{fa} varies as a function of η for $\epsilon = 0.0$ and 0.05. For a 0% tolerance level in MFD (i.e., ϵ=0), P_{fa} decreases gradually as η is increased. However, we notice that P_{fa} drastically decreases for all the source models when the tolerance level is increased to 5%. This indicates that P_{fa} is more sensitive to ϵ than η. For the rest of the experiments, we choose ϵ=0.05 and $b_{\text{TBF}} = 6000$.

Increasing η causes a delay in reporting misbehaving flows and may adversely impact the effectiveness of MFD. We examine this issue in Case 2. We set the value of γ (fraction of misbehaving flows) to be 0.1. Fig. 7a and 7b show the variation of P_{sd} and P_{fa} as η is increased from 1 to 10. The effect of η on P_{sd} for CBR source is minimal. For the other source models, P_{sd} decreases sharply when η is increased and the rate of decrease varies across the source models. From Fig. 7b, we can infer that only in the case of the EXP2 source model is P_{fa} sensitive to the value of η. With $\eta = 1$, we can detect most of the malicious flows with EXP1 (79%), CBR (83%) and PARETO (64%) sources with virtually zero P_{fa}. In the case for EXP2, there is a trade-off between maximizing P_{sd} and minimizing P_{fa} as we choose the value for η. This indicates that burstier sources are more difficult to detect.

We also measured the detection time for each correctly identified malicious flow and plotted the distributions in Fig. 7c. With $\eta = 1$, the average detection time is 26.9 seconds, which is less than 1/10 of the average duration of a flow. 90% of the flows are detected within 78.9 seconds. When

Source	EXP1	EXP2	CBR	PARETO
P_{sd}	1.0	1.0	1.0	1.0
P_{fa}	0.0077	0.027	0.012	0.0013
P_{mis}	0.003	0.00021	0.0072	0.0037

Table 1: **Case 3:** One large malicious flow and many small complying flows. $\eta = 5$, $b_{\text{TBF}} = 6000$, $\epsilon = 0.05$.

we increase η to 5, the average detection time increases to 33.8 seconds, which is still reasonably fast. The 90%-tile detection time is 80.4 seconds in this case.

The simulations in Case 2 show that basic results of Furies-MFD hold across different source models. Although long range dependent traffic like PARETO is harder to detect, we can achieve a reasonable success rate (0.64) with zero false alarms. The presence of burstier sources (EXP2) pose challenges to the Furies-MFD scheme, and we need to choose the value for η carefully to maximize P_{sd} while keeping P_{fa} reasonably small.

We have so far considered only homogeneous flows with the same rates. In the next two cases, we consider an extreme case where there is one large flow with peak rate r_{large} and many small flows with peak rate r_{small}, where $r_{\text{large}} = 10 \times r_{\text{small}}$. We model the small flows the same way as in Case 1 and repeat our experiments for four different source models. In Case 3, only the large flow is misbehaving. The results are summarized in Table 1. For all source models, we always successfully detect the misbehaving large flow and P_{fa} is less than 0.03.

Case 4 presents a more interesting scenario where the large flow is complying while a small fraction of the small flows are misbehaving. The probability of a misbehaving flow is γ. Intuitively, we suspect that detection is harder in this case, because the misbehaving flows can hide in the aggregate traffic by stealing the idle bandwidth allocated to the large flow. Since the traffic policer can only enforce the total allocated rate, the malicious flows may not be detected. Fig. 8 shows the P_{sd} achieved for

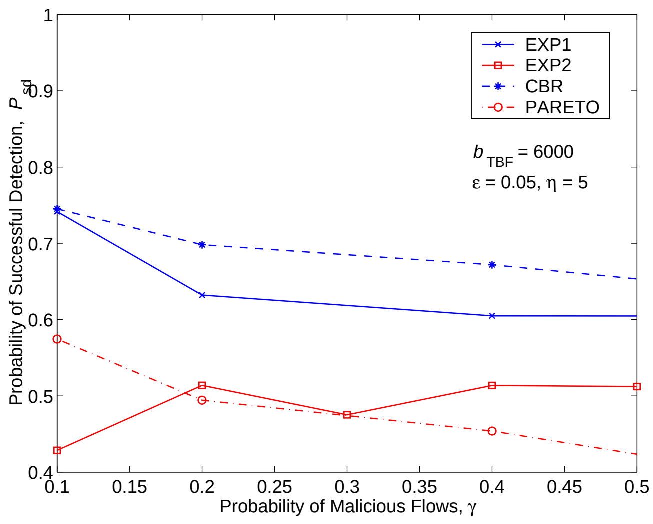


Figure 8: **Case 4:** One large flow (r_{large}) and many small flows (r_{small}); γ of small flows misbehave.

Source Model	EXP1	EXP2	CBR	PARETO
$\gamma = 0.1$				
P_{sd}	0.74	0.43	0.75	0.57
P_{fa}	0.00066	0.011	0.0	0.0
P_{mis}	0.0032	0.00016	0.036	0.0028
$\gamma = 0.5$				
P_{sd}	0.61	0.51	0.67	0.39
P_{fa}	0.00067	0.025	0.0	0.0
P_{mis}	0.0030	0.00047	0.050	0.0022

Table 2: **Case 4:** One large flow and many small flows. A fraction, γ, of small flows misbehave. η = 5, b_{TBF} = 6000, $\epsilon = 0.05$.

Source Model	HET	EXP1	EXP2	CBR	PARETO
$\eta = 1$					
P_{sd}	0.55	0.79	0.73	0.83	0.64
P_{fa}	0.0	0.0	0.14	0.0025	0.0
P_{mis}	0.0030	0.0028	0.00016	0.0079	0.0031

Table 3: Comparisons between heterogeneous and homogeneous source models: $\gamma = 0.1$, b_{TBF} = 6000,$\epsilon = 0.05$.

different values of γ and Table 2 summarize P_{sd}, P_{fa} and P_{mis} for $\gamma = 0.1$ and 0.5. Surprisingly, we notice that the P_{sd} achieved with $\gamma = 0.1$ for EXP1, CBR, and PARETO sources are fairly close to the results in Case 1, where there is no large flow. But for EXP2, the success rate is significantly smaller (P_{sd}=0.43 in Case 4 as supposed to 0.54 in Case 1). When γ increases, the success rate P_{sd} decreases for EXP1, CBR and PARETO source models. With EXP2, P_{sd} fluctuates as we vary γ, and is actually higher at γ=0.5 than γ=0.1. This is because the active cycle of EXP2 is very short (10%), and can easily go undetected if it coincides with the idle period of the large flow. However, when the fraction of malicious flows increases, there is an increased likelihood that some of the malicious flows will synchronize or overlap in their active cycles, leading to overflow of the TBF at the traffic policer. When the aggregate rate is violated, all the flows sharing the same subfield ($FidIn$ or $FidEg$) will be monitored individually (micro-monitoring) and the malicious flow can be correctly identified. The probability of false alarms P_{fa} and mis-marked packets P_{mis} are negligible in this case across different values of γ and source models.

So far, we have been using only homogeneous flows in our simulations. The next experiment uses a mix of the four different source models (EXP1, EXP2, CBR and PARETO), each with different peak rates, idle times and burst times. Each arriving flow chooses among these source models at random. This heterogenous source is denoted as HET. We repeat the experiment in Case 2, with η=1 using HET, and compare the results with Case 2 where homogeneous flows are used. Results are summarized in Table 3. With HET source, the success rate P_{sd} is lower than all the other homogeneous source models, but the differences in P_{fa} and P_{mis} are negligible. It is difficult to tune the hysteresis parameters to optimize the overall performance. This is the inherent limitation of using token bucket filters as traffic policers in Furies.

6 Implementation and Prototyping

In this section, we provide a brief description of how we implemented the Furies architecture on top of the Click modular router [8]. We extended the Click router to support all the traffic policing and admission control functionalities of our architecture. Using this implementation, we measured the performance overhead incurred at an edge router by adding the policing and monitoring tools of Furies. The current implementation works on Linux 2.2.16 and 2.2.17 kernels. Our architecture has been operationally verified in our laboratory's test-bed. We will provide an overview of the implementation and performance measurements in sections 6.1 and 6.2.

6.1 Overview of Implementation

Click is a new software architecture for building flexible and configurable routers developed by Kohler et al. [8]. A Click router is assembled from packet processing modules called elements. Individual elements implement simple router functions like packet classification, queuing and scheduling.

We implemented different *elements* that perform traffic policing and admission control and integrated them with the existing elements of Click. Since the configurations of Click are modular, we found it very easy to extend Click to include Furies.

We have added two main elements to Click: the reservation agent (RA) and the monitoring agent (MA). The reservation agent is responsible for receiving flow requests and forwarding them to the Resource Manager (RM) of FURIES. The RA is also responsible for forwarding responses from the Resource Manager to the client. The MA handles the traffic monitoring and policing of admitted flows, i.e., the functionalities of the TMs and TPs in the Furies architecture (Section 2.2) The MA is responsible for: estimating the traffic demands across edge routers, detecting and penalizing misbehaving flows, and continuously updating the Resource Manager of the aggregate traffic demand distributions.

The communication between the Click router and the Resource Manager is performed through SNMP. In order to enhance the throughput of the Click router, we reduce the number of context switches required for processing the control packets from the RM. We achieve this by batching the messages from the RM to the Click router.

6.2 Performance Evaluation

We used our implementation to evaluate certain performance metrics which could not be accurately quantified through simulations. One such important metric was the overhead of implementing the RA and MA in an edge router. To obtain a reasonable and realistic picture of this overhead, we compared the maximum throughput obtained from our implementation to a basic implementation of Click which did not contain any of the monitoring tools (hereafter referred to as default Click). A quantification of this overhead was necessary to determine whether it is practical to deploy FURIES. We also measured the time required by FURIES to admit a flow as a function of the load on the network. This time is helpful in quantifying the response time for performing admission control on a new flow request.

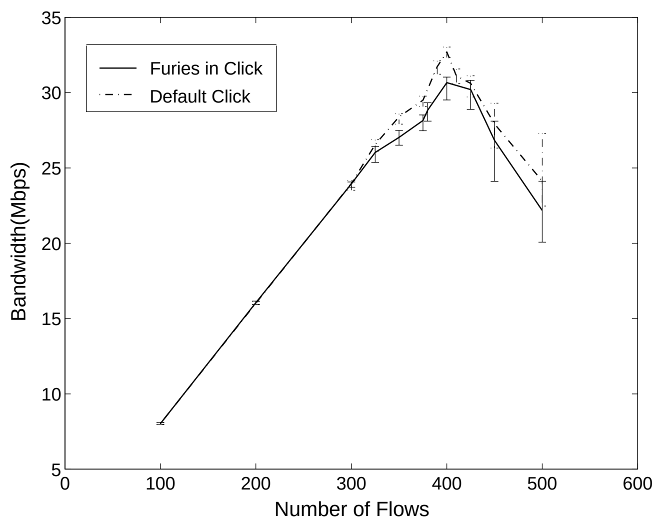


Figure 9: Comparison of Peak Throughput of Furies+Click over Default Click

6.2.1 Experimental Setup and Methodology

For evaluating the performance, we set up our own cluster of machines over a 10.2.2.0/24 network. The experimental setup consisted of a total of six Intel PCs running Linux. A Pentium-III 650Mhz machine was used as the Click router. This machine used a 3com 3c905 100Mbps Ethernet controller and had 256 KB of L2 cache. Another machine, a Celeron-400 Mhz with a 128 KB of L2 cache was used as one of the traffic generators. The other traffic generators were Dell 6350, 4-processor 650 Mhz machines. We use two more machines as Sink and Resource Manager. All these machines are connected to a backbone router using 100 Mbps connections. The router is a Bay Networks Accelar-1100B router with the capacity to support 16 100 Mbps Ethernet ports.

To make our measurements more realistic, we used an IP routing table from a BBN planet edge router [20]. The routing table contained 43,872 entries. We disabled the IP-lookup cache in order to characterize lookup time and compared that time with the time taken for traffic monitoring. The traffic distribution was periodically sent to the RM every 100ms. We modified Mgen [21], a publicly available traffic modeling software, to generate traffic for our experiments.

Other than this implementation, we also built a proof-of-concept prototype on a powerful computational testbed in our laboratory. We verified all the functionalities of our architecture by setting up a virtual network of seven ingress-egress network over this testbed.

6.2.2 Experimental Results

In our first experiment, we measured the maximum throughput of our implementation at different loads and compared it to a default Click router. We tested our implementation for varying loads. A basic flow in our setup had a bandwidth 80kbps and a packet size of 1024 bytes. We varied the number of flows to generate varying loads. As the number of flows increased, the amount of policing and state needed at the edge router also increased.

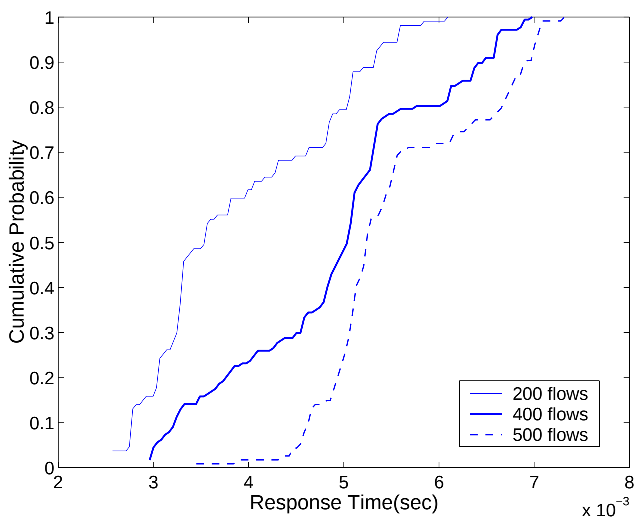


Figure 10: Response Time of Flow Allocation for varying loads

Figure 9 shows a comparison of the throughput of our Click implementation with the monitoring tools and the default Click. From the figure, we can infer that the percentage overhead (percentage difference in throughput) is negligible for a large range of loads. The maximum percentage difference observed in our experiments was 5%. This occured at a load of 400 flows. The default Click gave a peak throughput of 32 Mbps while our implementation gave 30.5 Mbps. At higher loads, the throughput of both decreased.

In the second experiment, we measured the response time for flow allocation at varying loads. We maintained a particular load in the system by maintaining a constant number of active flows and sent dummy flow requests to the Click router from the Traffic generator. There are three stages in the process of obtaining a response for a flow request: the RA in Click forwards the request to the RM, the RM performs admission control on the flow and sends the response to the requesting entity through the RA.

In Figure 10, we plot the cumulative distribution of the response time at three different loads. We specifically chose three characteristic values of load - 200 flows (small load), 400 flows (saturation point) and 500 flows (very high load). From the graph, we can observe that the mean response time increases as the load increases and the CDF shifts to the right. In all our experiments, we observed a minimum response time of 2.5 ms and a maximum of 7.2 ms. Our results indicate that the standard deviation of our response time is high. This can be attributed to the batching of responses at the RM and timer-based processing of flow requests at the Click router. These two routines increase the variance of the response time for flow requests. This variance is tolerable since the mean response time is small.

6.3 Discussions

We found it extremely easy to build extra router mechanisms on top of Click. Our entire implementation consists of 4463 lines of C++ code and the effort taken to integrate our code with Click was minimal. Our experimental results indicate that the performance overhead incurred by adding our monitoring tools to Click router is less than 5% at peak throughput. At smaller loads, the overhead is negligible.

We performed all our measurements in the user and kernel modes of Click. In our implementation, we observed a negligible difference in performance between the interrupt-driven kernel mode and the interrupt-driven user modes of Click. However, we did not use the *Poll-device* element optimization of Click. Our implementation also used a much larger routing table than the one used in [8]. Studies in [22, 23] have shown that increasing the size of the routing table can adversely affect the throughput of the system. Click does not support many optimizations for fast table lookups. The performance of Click has also been optimized for DEC 21140 Tulip 64-bit PCI controllers and our setup used a 32-bit 3com Ethernet controller. Though these issues may affect the net throughput of the system, we believe that it will not change the percentage difference of the throughput.

7 Deployment Issues

This section describes some of the issues involved in deploying Furies in the existing Internet.
Changes to Routers: To deploy Furies, all the routers must support at least simple priority scheduling. No changes are required in the core routers, while the policing and admission control software need to be added to the edge routers. The Resource Manager module can be added to control points of the various POPs in an ISP. From our implementation experience, we infer that the overhead of implementing Furies in an edge router is minimal.
Virtual Private Networks: Furies can be deployed within an ISP to support Virtual Private Network (VPN) customers. The mechanisms of Furies can be applied to provide an abstraction of a VPN overlay network by treating the VPN endpoints as IE-Pipes. We do not need to change any of the underlying admission control or traffic policing mechanisms.
Multiple ISPs: If indeed Furies is deployed, its deployment will be incremental and not all ISPs will be Furies-enabled. A flow that passes through multiple ISPs may not receive the best performance guarantees if some intermediary ISPs do not support Furies. Furies can be extended to support a *Confederation* of ISPs, which is a group of ISPs that coordinate among each other to set up a larger virtual ISP through peering agreements. Every peering point is associated with a set of SLAs, and a flow that is admitted in one ISP can be guaranteed its level of service within the confederation. Furies also provides a way for sharing resources across ISPs and does not assume any trust relationships between ISPs.

8 Related Work

Providing better QoS assurance as offered by stateful networks like Int-Serv, while maintaining the scalability of a stateless network architecture like Diff-Serv, the Furies architecture performs admission control only at the edge router. Furies does not perform active probing nor make any admission control decisions based on direct congestion detection. Our admission control decisions are based on *passive* measurements of existing traffic rather than the worst case bounds describing the flows, as in *parameter-based* admission control. Many algorithms and principles outlined in the *measurement-based* admission control (MBAC) literature apply in our work, and we definitely benefit from results in [15, 24, 25], to name a few. We chose measurement-based over parameter-based for two reasons. First, MBAC yields higher network utilization, and secondly it is difficult to describe Internet traffic with such diversity and unpredictability with a reasonably small set of parameters.

Reference [26] proposed a service model, called *hose*, which specifies the capacity required for aggregate traffic from one endpoint to the set of other endpoints in the VPN customer sites. Each *hose* is associated with a performance guarantee. Furies follows the same fundamental idea, but uses a different abstraction. We infer traffic demands from direct flow-level measurements using the same methodology described in [10], which was successfully applied to a large, operational ISP network. We then use this knowledge to construct a traffic matrix that captures the bandwidth requirement between every pair of ingress-egress ARs, i.e., *IE-Pipe*. The admission threshold is determined based on this traffic matrix so that the link bandwidth is allocated to the competing IE-Pipes in proportion to their traffic demands.

Terzis, et al. defined the primitives of a *two-tier* resource allocation model in [6]. A simplified RSVP is used as an intra-domain resource allocation protocol for aggregate traffic between access routers, and admission control is performed on a hop-by-hop basis. The bandwidth broker (BB) in each domain is responsible for establishing bilateral resource agreements using a proposed inter-domain protocol. Although Furies adopts the fundamental principle of the *two-tier* model for intra- and inter-domain resource allocation, its mechanisms and emphasis are significantly different from [6] or other BB literature [3, 4]. First, admission control is only performed by the Resource Manager (RM), without propagating the reservation request from end-to-end. Secondly, resource allocation decisions are made based on the estimated traffic demand and global knowledge of intra-domain topology, instead of a hysteresis process as detailed in [6]. Finally, Furies proposes a scalable approach to perform traffic policing and detect misbehaving flows, which are not addressed in the previous BB literature.

The most relevant work with respect to our traffic policing mechanism is Stochastic Fair Blue (SFB) proposed by W. Feng, et al. in [12]. SFB provides a scalable way to identify and rate-limit non-responsive flows using BLUE, which marks or drops packets based on loss rate and link utilization history. The goal of SFB is to manage congestion, while Furies attempts to identify misbehaving flows based on utilization history. The idea of classifying good versus bad (non-responsive in SFB or misbehaving in Furies) flows is similar, but the associated algorithm is different.

9 Conclusion

In this paper, we developed Furies, a framework for scalable traffic policing and admission control, which can be incrementally deployed in the current Internet. Though detection of malicious flows has long been recognized as an important component, a practical and scalable way of implementing it has not been studied in great detail. Furies only requires $O(\sqrt{n})$ state information to be maintained at edge routers, which is substantially better than previous approaches.

In our simulations, we tested our admission control and malicious flow detection schemes using a variety of source models. The basic results hold across different extreme cases and therefore justify the robustness of Furies. However, we found it hard to detect long-range dependent and bursty malicious sources. Our implementation experience indicates the practicality and ease of deployment of Furies.

References

[1] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, "ReSerVation Protocol (RSVP) version 1 functional specification," Internet RFC 2205, IETF Network Working Group, September 1997.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weise, "An architecture for differentiated services," Internet RFC 2475, IETF Network Working Group, December 1998.

[3] Internet2 QoS Working Group: Qbone testbed, `http://www.internet2.edu/qos/qbone/`.

[4] B. Stiller, T. Braun, M. Günter and B. Plattner, "The CATI project: charging and accounting technology for the Internet," *Proc. of 4th European Conference: Multimedia Applications, Services and Techniques*, pp. 281-96, May 1999.

[5] R. Rajan, D. Verma, S. Kamat, E. Felstaine, and S. Herzog, "A policy framework for integrated and differentiated services in the Internet," *IEEE Network Magazine*, vol. 13, no. 5, pp. 36-41, September/October 1999.

[6] A. Terzis, L. Wang, J. Ogawa and L. Zhang, "A Two-Tier Resource Management Model For The Internet," *Global Internet*, pp. 1808-17, December 1999.

[7] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," *Proc. of ACM SICGOMM*, pp. 81-94, September 1999.

[8] E. Kohler, R. Morris, B. Chen, J. Jannotti and M. F. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Systems*, vol. 18, no. 4, November 2000.

[9] A. Sridharan, S. Bhattacharyya, C. Diot, R. Guerin, J. Jetcheva and N. Taft, "On the Impact of Traffic Aggregation on Routing Performance," June 2000. (submitted for publication)

[10] A. Feldman, A. Greenberg, C. Lund, N. Reingold, J. Rexford and F. True, "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience," *ACM Proc. Sigcomm*, pp. August 2000.

[11] D. Verma, *Supporting Service Level Agreements on IP Networks*, Macmillan Technical Publishing, 1999.

[12] W. Feng, D. D. Kandlur, D. Saha and K. G. Shin, "BLUE: A New Class of Active Queue Management Algorithms," *U. Michigan Technical Report CSE-TR-387-99*, April 1999.

[13] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *Journal of High Speed Networks*, vol.3, no.4, pp. 389-412, 1994.

[14] L. Breslau, E. W. Knightly, S. Shenker, I. Stoica and H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance," *ACM Proc. Sigcomm*, pp. August 2000.

[15] S. Jamin, P. Danzig, S. Shenker, and L. Zhang, " A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks," *IEEE/ACM Trans. in Networking*, vol.5, no.1, pp. 56-70, February 1997.

[16] C. Fraleigh, S. Moon, C. Diot, B. Lyles and F. Tobagi, "Architecture of a Passive Monitoring System for Backbone IP Networks," October 2000. (submitted for publication)

[17] ITU-T Recommendation P.59, "Artificial conversational speech," 1993.

[18] W. Willinger, M. Taqqu, R. Sherman and D. Wilson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *ACM Proc. Sigcomm*, pp. 100-113, August 1995.

[19] M. Crovella and A. Bestavros, "Self-Similarity In World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Trans. on Networking*, vol.5, no.6, pp. 835-46, December 1997.

[20] IP Routing tables, `http://www.merit.edu/~ipma/tools/lookingglass.html`.

[21] The Naval Research Laboratory (NRL), "Multi-Generator" (MGEN) Toolset, `http://manimac.itd.nrl.navy.mil/MGEN/`.

[22] M. Degermark, A. Brodnix, S. Carlsson and S. Pink, "Small forwarding tables for fast routing lookups," *ACM Proc. Sigcomm*, pp.3-14, October 1997.

[23] M. Waldvogel, G. Varghese, J. Turner and B. Plattner, "Scalable high speed IP routing lookups," In *Proceedings of ACM SIGCOMM Conference(SIGCOMM '97)*, pp 25-38, October 1997.

[24] M. Grossglauser and D. Tse, "A Framework for Robust Measurement-Based Admission Control," Proc. ACM Sigcomm, vol. 27, no. 4, pp. 237-248, September 1997.

[25] E. W. Knightly and J. Qiu, "Measurement-based Admission Control with Aggregate Traffic Envelopes," *IEEE ITWDC*, September 1998.

[26] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan and J. E. Van der Merwe, "A flexible model for resource management in virtual private networks," *Proc. of ACM Sigcomm*, pp. 95-108, September 1999.