GREEN: A TCP Equation-Based Approach to Active Queue Management *

Apu Kapadia†
Dept. of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin
Urbana, IL 61801
akapadia@uiuc.edu

Wu-chun Feng
Computer & Computational Sciences Division
Los Alamos National Laboratory
Los Alamos, NM 87545
feng@lanl.gov

Roy H. Campbell
Dept. of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin
Urbana, IL 61801
rhc@uiuc.edu

ABSTRACT

GREEN is a stateless queue-management algorithm that removes TCP's bias against connections with longer round-trip times, resulting in a high degree of fairness while maintaining high link utilization, low packet loss, and short queue sizes. GREEN applies the knowledge of the steady-state behavior of TCP connections to drop packets proactively, thus preventing long-lived TCP flows from ever inducing congestion. It also prevents shorter round-trip time (RTT) flows from grabbing more than their fair share of bandwidth. Consequently, GREEN ensures much higher fairness between flows than other queue-management schemes.

Initial work on GREEN evaluated the performance of an ideal GREEN router, which was assumed to have global knowledge of each individual flow's RTT. Furthermore, the algorithm was susceptible to severe under-utilization of the link bandwidth in the presence of short-lived or low-bandwidth flows. To address these shortcomings, we present an incentive-based solution where flows benefit by embedding their current RTT estimates in the TCP header. We then present results for an IDMaps-based solution where the router estimates a flow's RTT by performing lookups in a local IDMaps database. Simulation results demonstrate that our algorithm is resilient for both long-lived as well as short-lived and low-bandwidth connections.

*This work was supported by the U.S. Department of Energy through Los Alamos National Laboratory contract W-7405-ENG-36 with funding provided by the DOE Laboratory-Directed Research and Development Program. This paper is available as Los Alamos Unlimited Release technical report (LA-UR-04-1017) as well as UIUC technical report (UIUCDCS-R-2004-2408/UILU-ENG-2004-1710).

†Apu Kapadia was funded in part by the U.S. Dept. of Energy's High-Performance Computer Science Fellowship through Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratory.

1. INTRODUCTION

Network congestion induced by TCP traffic leads to lost packets, wasting all the resources that the packet consumed on its way from source to destination. Active queue-management (AQM) schemes such as RED [8], Blue [4], AVQ [12], REM [1], and PI [10] actively detect congestion early and appropriately react to impending congestion that would otherwise fill the queue and cause a burst of packet drops. REM, PI, and AVQ are designed to increase the link utilization at a router while maintaining small queue sizes. However, they do not provide adequate fairness, which is sacrificed at the cost of higher utilization. While RED and Blue were designed to stabilize queue sizes at low levels, Flow Random Early Drop (FRED) [14] and Stochastic Fair Blue (SFB) [5] improve on their performance by operating at the flow level. FRED and SFB attempt to enhance throughput-fairness between flows by penalizing flows of higher bandwidth, but we show that these approaches do not perform well with flows that have widely varying RTTs.

GREEN [6] is a *proactive queue management (PQM)* scheme which regulates TCP flows over the same link to a fair sending rate, and hence, prevents them from inducing congestion. GREEN does so by using the knowledge of TCP's steady-state behavior. Feng et al. [6] present promising preliminary results for a GREEN router, which exhibits high fairness with flows of widely varying RTTs. However, their design suffers from severe under-utilization in the presence of short-lived or low-bandwidth flows. Furthermore, the authors assume global RTT knowledge of flows at the router. In this paper, we provide a practical algorithm for GREEN and show that our approach achieves high link utilization in the presence of short-lived or low-bandwidth flows. We propose stateless RTT estimation techniques for GREEN and show how a high degree of fairness is achieved. Hence, our solution is practical for realistic deployment and overcomes several limitations of GREEN, as proposed in [6].

GREEN's congestion-preventing, proactive, queue-management scheme is based on a mathematical model of

the steady-state behavior of TCP's congestion-avoidance algorithm [15]. GREEN ensures that well-behaved TCP flows can be regulated to receive their fair share of the link bandwidth while simultaneously maintaining high link utilization and low packet loss. However, this mathematical model relies on the ability of the router to infer the RTT of a flow. Feng et al. [6] presented results for an ideal GREEN router that was assumed to have access to RTT information and the number of active flows. We propose a practical solution for GREEN based on two RTT estimation techniques: (1) embedded RTTs where senders include current RTT estimates within the TCP header and (2) the use of a service like IDMaps [9], an Internet host-distance estimating service. We also include an estimator for the number of flows. While IDMaps does not provide exact RTT estimates, we examine the impact of an IDMaps-based solution and show that even in the face of moderately inaccurate RTT estimates, GREEN still outperforms other flow-based AQM schemes, such as FRED and SFB, in terms of fairness, while still maintaining high link utilization and low packet loss. We also discuss the impact of short-lived and low-bandwidth flows and show how our algorithm for GREEN is able to perform well under such conditions. All our results are based on simulations using the *ns-2* [17] simulator.

Our main contribution is to provide an algorithm for GREEN that is practical to deploy and maintains high utilization and fairness in the face of flows with widely-varying RTTs and in the presence of low-bandwidth and short-lived connections.

The rest of the paper is organized as follows. In Section 2 we present the basic algorithm for GREEN. Section 3 briefly discusses the placement of GREEN routers. In Section 4 we discuss various AQM techniques and motivate the comparison of GREEN with FRED and SFB. Section 5 describes RTT estimation techniques for GREEN based on RTT information being embedded within the header and based on IDMaps. Section 6 describes our experimental setup for comparing the various queue-management schemes. In Section 7 we evaluate our implementations of GREEN FRED, SFB, and Droptail using three sets of experiments. Section 8 briefly discusses GREEN's state and computational requirements, and we finally conclude in Section 9.

2. ALGORITHM

GREEN applies knowledge of the steady-state behavior of TCP connections at the router to drop (or mark) packets intelligently for congestion notification. By using such a mechanism, a router can give each connection its fair share of bandwidth. The throughput of a TCP connection depends, among other factors, on its round-trip time (RTT) and the probability that its packets are dropped in the network. Specifically, Mathis et al. [15] show that a connection's throughput satisfies the following equation under certain simplifying assumptions:

$$BW = \frac{MSS \times c}{RTT \times \sqrt{p}} \quad (1)$$

where BW is the bandwidth or throughput of the connection, MSS is the maximum segment size, RTT is its round-trip time, and p is the packet-loss probability for that connection. The constant c depends on the acknowledgment strategy that is used (i.e., delayed or every packet) and whether packets are assumed to be lost periodically or at random.

In general, this model may not be applicable in environments where there are sustained multiple packet losses for a flow within a single RTT (causing repeated timeouts). This model may also not apply to very short connections that never reach steady state, or to connections whose window sizes are artificially limited by the receiver's flow-control window. For our analysis, we first assume that all connections satisfy the assumptions required for this model. Specifically, our simulations focus on long-lived FTP connections that are able to attain their steady-state bandwidths. We then relax this assumption and show how this model can be used even in the face of low-bandwidth and short-lived connections.

Now, let us consider a scenario where there are N active flows at a router on a particular outgoing link of capacity L. In GREEN, an active flow is a TCP source that has outstanding data to be sent. If a flow has had at least 1 packet go through the router within a certain interval, or *window*, of time, GREEN assumes that the flow is still active. The fair-share throughput of each flow is L/N (assuming each source attempts to transmit at least at that rate). Substituting L/N for BW in Equation (1), we derive the following expression for loss probability p:

$$p = \left(\frac{N \times MSS \times c}{L \times RTT}\right)^2 \quad (2)$$

By using this value of p as the drop probability for congestion notification, GREEN "coerces" flows into sending at their fair rate. Note that GREEN applies this marking probability to *all* arriving packets, where the value of p depends on the flow. Because p depends on the number of flows and the round-trip time of each flow, congestion notification is more aggressive for large N and small RTT. By including RTT as an inverse parameter in the equation, GREEN eliminates the bias of favoring TCP connections with smaller RTTs with respect to throughput [13]. (Recall that TCP connections with smaller RTTs can increase their window size faster due to the smaller RTT and are therefore more aggressive. Hence, these flows grab more than their fair share of bandwidth, which leads to this bias.)

This drop policy gives extremely good results in terms of fairness when all flows are long-lived, high-bandwidth flows (e.g., FTP). But when there are low-bandwidth flows present (e.g., there may be several 56-Kbps dial-up connections through the router), these connections cannot use their "fair share" as allotted by GREEN. As a result, utilization suffers since low-bandwidth flows do not use their entire fair share of bandwidth. To compensate for this, we first observe that the low-bandwidth connections are using less than their fair share or are bottlenecked at some other link. Let the remaining available bandwidth (after accounting for the aggregate bandwidth used by the low-bandwidth flows) at the GREEN router be L'. Let the remaining number of flows be N'. We would like to allocate L'/N' as the fair share to the remaining flows. If we recalculate drop probabilities based on these

values, then we can utilize the remaining bandwidth at the GREEN router. In reality, it is difficult to estimate these values without maintaining per-flow state. Instead, we observe that $N'/L' = \gamma(t)N/L$, where $\gamma(t)$ is some constant at time t. We propose a simple negative feedback control algorithm to estimate the value of $\gamma(t)$ at time t and show how GREEN performs extremely well; this adaptation parameter results in high utilization of the link while providing high fairness between flows. Hence, we propose an approach that corrects for previous limitations of GREEN and show how GREEN can indeed be deployed practically. The updated equation for p is:

$$p = \left(\frac{N \times MSS \times c}{\gamma(t) \times L \times RTT}\right)^2 \quad (3)$$

Using this equation, GREEN can scale (up or down) the sending rates of senders to adjust the overall link utilization. Moreover, $\gamma(t)$ has a negligible effect on short-lived and low bandwidth flows.

We use the following increase and decrease policies to update estimates of $\gamma(t)$. GREEN keeps track of the current link utilization by counting the number of departures within a given *window* of time. This is the same window that GREEN uses to estimate the number of flows. For simplicity, we use a window of one second. If the current utilization $U(t)$ is 50%, for example, then we effectively need to scale the bandwidth of each flow by a factor of 2, which would theoretically yield 100% utilization of the link (assuming that each flow is capable at sending at their fair rates). In this case, $\gamma(t+1) = 2\gamma(t)$. However, to avoid oscillations, we grow $\gamma(t)$ at a slower rate and aim to increase utilization halfway between the current utilization and full utilization. In our example, we would attempt to reach a target utilization of 75% (scaling bandwidth by 1.5 for each flow) yielding $\gamma(t+1) = 1.5\gamma(t)$, or in general $\gamma(t+1) = \frac{1+U(t)}{2U(t)}\gamma(t)$. We use this policy when $U(t) < 0.98$, and hence, GREEN reacts to low utilization by increasing sender rates and hence overall utilization. If there are packet drops at the link that are induced by full buffers, this is an indication of congestion and GREEN reacts by reducing $\gamma(t)$ by a multiplicative factor of 0.95, that is $\gamma(t+1) = 0.95\gamma(t)$. We use this value to reduce any wide oscillations in link utilization. Our experiments show that GREEN maintains high utilization using these control parameters and reserve further control-theoretic analysis for future work.

The pseudocode for the GREEN algorithm is shown below.

```
Enqueue(Packet pkt)
  RTT ← obtainRTT(pkt)
  p ← (N×MSS×c / γ×L×RTT)^2

  u ← UniformRand(0,1)
  if (u ≤ p) then
    drop(pkt)
  else
    addToQueue(pkt)
  end if

  if (currentTime() − lastUpdate ≥ window) then
    update(currentUtil, N, queueDrops)
    lastUpdate ← currentTime()

    if (queueDrops > 0) then
      γ ← 0.95γ
    else if currentUtil < 0.98 then
      γ ← (1+currentUtil / 2×currentUtil) γ
    end if
  end if
```

When a packet is received at the GREEN router, GREEN first obtains the packet's RTT (we describe our approaches for doing so in Section 5). p is the drop probability as calculated by Equation (3). u is a random number selected over a uniformly-distributed interval $[0, 1]$. If $u \leq p$ then the packet is dropped (i.e., each packet is probabilistically dropped with the calculated probability p), and otherwise it is added to the outgoing queue. If the *window* of time has elapsed for updating the values of current link utilization *currentUtil*, the number of active flows N, and the number of queue drops due to overflow *queueDrops*, then these values are updated. Additionally, the value of γ is adjusted based on *currentUtil* and *queueDrops*.

3. ISSUES AND PLACEMENT OF GREEN ROUTERS

Since GREEN calculates p, the drop probability for a flow, GREEN routers cannot be "composed" since this would alter the overall drop probability for a flow. Hence GREEN is mainly suited as an edge router, where organizations can enforce fairness between flows leaving the organization through a bottleneck link. This ensures that the GREEN drop probability is applied to a flow only once. For flows that are bottlenecked elsewhere, the overall drop probability for the flow is not adversely affected by the action of a single GREEN router. Another ideal application of GREEN would be the outgoing link of an organization with popular FTP servers, where there are several competing TCP connections, which are long-lived and have varying RTTs. Another justification for keeping GREEN routers at the edge is that shorter-RTT flows, which cause unfairness at a link, are effectively limited closer to the source. Routers in the core of the network usually carry longer-haul flows, where GREEN's algorithms may not be too effective. Having GREEN routers limit shorter-RTT flows closer to the source is more desirable since this affects flows closer to the source. We hope to validate this claim more rigorously in future work.

4. RELATED WORK

End-to-end schemes have been proposed to correct for TCP's bias against longer RTT flows by requiring TCP senders to increase their congestion windows by a constant proportional to the square of the RTT [7][19]. However, these schemes rely on a *window constant* that is hard to calculate and varies with the topology of the network. Moreover, these changes affect TCP's algorithm, and hence, these changes are not trivial. In this paper we focus on developing a practical and deployable solution. Thus, we examine GREEN's performance under a packet-dropping model (as opposed to a marking model, which relies on ECN-capable endpoints) and use TCP Reno at the endpoints of our simulations. Such a solution would only require a GREEN-capable router; minimal or no modifications to the endpoints

would be necessary.

Since RED [8] was not designed to discriminate between flows, RED applies the same loss rate to all flows irrespective of their bandwidths. Flow-RED (FRED) [14] attempts to remedy this "unfairness" by applying a loss rate to a flow that is based on its queue-occupancy statistics. Hence, higher bandwidth flows that consume more resources at the router receive a higher loss rate.

Because RED relies only on queue-occupancy statistics, maintaining a stabilized queue is difficult without correct parameterization. Blue [4] proposes a different approach, which uses link utilization and packet-loss history to manage congestion. In particular, Blue increases the marking probability when packet losses increase and decreases the marking probability when link utilization decreases. However, Blue does *not* discriminate between flows and applies this marking probability to all flows, irrespective of their bandwidths.

Stochastic Fair Blue (SFB) [5] tries to correct the above problem by applying a Bloom filter [2] to hash flows into L levels of N bins. Each bin maintains queue-occupancy statistics for flows that map into that bin and a corresponding drop probability p_m. For a given flow, SFB calculates the marking probability based on queue-occupancy statistics of the various bins. This approach is effective in applying a more aggressive marking probability to flows of higher bandwidth and a low marking probability to flows of lower bandwidth. In sum, FRED and SFB are two recent AQM approaches that attempt to ensure bandwidth-fairness between flows.

In contrast to the approaches described above, more recent AQM schemes apply a control-theoretic view to increase the link utilization at a router. Random Exponential Marking (REM) [1] relies on REM-capable routers that support explicit congestion notification (ECN). Congestion measures are inserted into packets by these routers, and the overall marking probability is calculated based on this. This results in high utilization and low delay at the queues. The Proportional Integral (PI) [10] controller attempts to regulate the steady-state value of queue-size. The authors argue that PI has better theoretical properties and performance than RED and can stabilize queue sizes at low levels. Adaptive Virtual Queue (AVQ) [12] uses a *virtual queue* with a capacity less than the capacity of the link. Packets in the actual queue are marked when the packet induces an overflow in the virtual queue. Following this, the link capacities are recalculated. Such an approach can result in the desired utilization at a link. While AVQ's performance stability relies on a delay bound (d), explicit RTT information for flows is not exploited to provide throughput fairness. These approaches assume that all flows have the same RTT and experimental results are based on this assumption. Under the scenarios that we test, these assumptions do not hold, and fairness suffers. Therefore, we believe that GREEN's main contribution is to provide high utilization and fairness in the face of flows with widely varying RTTs (as well as in the presence of short-lived or low-bandwidth flows).

While the above approaches are important in that they increase the link utilization at a router and provide queue stability, they do not discriminate between flows and hence cannot ensure bandwidth fairness. In terms of bandwidth fairness, we expect the behavior of such schemes to be similar to that of Droptail (*FIFO* queueing with finite buffer) or RED, so we do not compare GREEN to PI, REM, or AVQ. Instead we focus on FRED and SFB since they were designed to provide a higher degree of fairness between flows. Future work could include applying the GREEN algorithm to AQM schemes designed for higher utilization. For example, when such an AQM scheme decides to mark a packet, a packet from the queue could be picked based on probabilities calculated by GREEN. This would enhance both the utilization *and* fairness at a router.

The scheme most similar to GREEN is XCP [11]. XCP requires significant changes to TCP implementations and requires a more complicated algorithm. However, like GREEN, XCP does suggest the use of embedded RTT in packet headers, amongst other parameters. In this paper, we present a simple and practical solution based on embedded RTT headers (as well as an IDMaps-like lookup service) but with no major modifications to TCP implementations.

Lastly, CHOKe [18] attempts to enforce fairness by penalizing aggressive flows and dropping packets for flows that occupy more buffer space than other flows. This philosophy matches that of FRED's; however, this can only slow down flows *after* they have already stolen bandwidth from other flows. CHOKe uses a statistical technique to do this—it compares each incoming packet with a random packet in the queue. If they match, CHOKe drops both packets. Otherwise CHOKe applies the regular RED algorithm. We have noticed that this results in much higher overall drop probabilities for flows and results in very low utilization of the link. For our simulation setup where RTTs vary widely, CHOKe appears to perform worse than RED in terms of fairness and link utilization, and we omit any comparisons in our simulations.

5. ESTIMATING RTT FOR FLOWS

GREEN's excellent performance comes with a tradeoff—the router must be able to infer a flow's RTT. Feng et al. [6] presented preliminary results for a GREEN router where the RTT was *assumed to be known at the router*. There are several ways of observing flows at the router and determining their RTTs, but these approaches require the use of per-flow state. Assuming per-flow state, one could simply use Fair Queueing and get high fairness and utilization at the router. Hence, to provide any benefit over Fair Queueing, GREEN must not use any per-flow state. As a result, we address this limitation with two approaches: embedded RTTs and IDMaps.

5.1 Embedded RTTs

The first approach requires TCP senders to embed their current RTT estimates within the TCP header. This value of RTT is the ideal value since exact drop probabilities for a given bandwidth can be calculated for a flow. This requires that the TCP sender be modified. There are two issues involved with this. Firstly, the sender TCP agent can lie about its RTT by specifying much larger RTTs. This would result in lower GREEN drop probabilities, and thus, the

flow could get more than the flow's fair share of bandwidth. Katabi et al. [11] suggest a solution to this problem—such flows will induce buffer overflows, following which the router can observe flows for which packets were dropped. If the flows react (via congestion control) faster than their specified RTTs, then these flows can be labeled as malicious and dealt with accordingly. The second issue is that it is unreasonable to expect that all TCP senders will be modified for GREEN. This would make GREEN unusable from any practical standpoint. We suggest the following solution to take care of this problem. GREEN can maintain two queues—a *GREEN-queue* for *compliant* flows that embed their current RTT estimates in the headers and a *regular queue* for *non-compliant* flows that are unmodified TCP senders. The regular queue can use any other AQM policy or simply a Droptail policy. It is easy to estimate the number of flows of each type, and service bandwidths for each queue can be adjusted accordingly. For example, if there are N_1 compliant flows, and N_2 non-compliant flows, then the GREEN-queue is serviced at a rate of $\frac{N_1}{N_1+N_2}$ times the link bandwidth, and likewise, the regular queue is serviced at $\frac{N_2}{N_1+N_2}$ times the link bandwidth. This serves two purposes—it ensures that compliant flows receive their fair share of the link bandwidth while non-compliant flows compete (albeit unfairly) for the remaining bandwidth. It also provides an incentive-based approach for TCP senders to be incrementally modified. Unmodified senders do not gain any advantage, while modified senders are able to send at their fair rates. We leave such an implementation for future work, and for now, concentrate on improving the GREEN algorithm for maintaining high utilization and fairness at the link. We refer to GREEN using embedded RTTs as GREEN-Embedded.

5.2 Estimating RTT using IDMaps

IDMaps [9] is a proposed scalable Internet-wide service that aims to provide Internet distance estimates. For example, the authors have suggested that IDMaps can be used by hosts for nearest mirror selection. Such a service is also well suited to GREEN, which can obtain RTT estimates for flows using IDMaps. We propose an architecture where GREEN routers are part of the IDMaps framework, and therefore, can perform fast lookups in a local IDMaps database. We acknowledge that IDMaps is currently a theoretical service, meaning that GREEN cannot rely on IDMaps for actual deployment yet; in the interim, embedded RTTs can be used for practical deployment. In any case, our results show that GREEN performs well in the face of unreliable RTT estimates. What is important is that GREEN should treat relatively longer RTT flows differently from relatively shorter RTT flows. For IDMaps, we assume that the RTT estimate can be anywhere between 50% and 200% of the actual value and show how this assumption affects the operation of GREEN. This assumption is consistent with the goals of IDMaps, which aims to provide RTT estimates within a "factor of two" of the actual value.

We also note that there are other proposed techniques for estimating the RTT between two IP addresses. For example, coordinate-based techniques like GNP [16] can be used for mapping AS-AS distances. IP addresses can be mapped to their respective ASes for rough RTT estimates. However, since no real-world data exists for these approaches, we assume RTT-estimating performance close to that of IDMaps.

5.2.1 IDMaps—Architecture

Jamin et al. [9] argue that providing highly accurate delay estimates (within 5%, for example) is not feasible. Instead they aim to provide a scalable solution with existing technology to provide delay estimates that are accurate to within a factor of two. Jamin et al. propose the deployment of *tracers* in the Internet. Tracers maintain raw distances amongst themselves and address prefixes (AP). The use of APs, as opposed to actual IP addresses, makes this solution feasible, trading off accuracy for scalability. The delay between two IP addresses is estimated by calculating the sum of the delays between the two tracers closest to the two address prefixes and the tracer-AP delays.

5.2.2 GREEN using IDMaps

We propose a solution in which GREEN routers also perform the duties of tracers and exchange distance information with other tracers. We do not expect this to add much overhead to existing traffic from routing updates. Furthermore, since GREEN is an edge router, the delays from sources within the organization to the GREEN router will be fairly low. GREEN can perform fast lookups in the local IDMaps database to obtain RTT estimates for a flow, based on the destination IP addresses (since the source IP address is assumed to be within the organization). GREEN calculates the drop probability based on the *estimated RTT*.

The accuracy of IDMaps estimates is sensitive to the number of tracers and their placement on the Internet. Jamin et al. have evaluated several graph-theoretic approaches as well as simple heuristics. In general, the accuracy of estimates increases when tracers are closer to the AP's (Address Prefixes). As mentioned earlier, GREEN routers will be co-located with the APs of that organization, and hence, will result in more accurate estimates. We refer to GREEN using IDMaps as GREEN-IDMaps.

6. EXPERIMENTAL SETUP AND METRICS

Here we evaluate and compare the performance of GREEN-Embedded and GREEN-IDMaps with respect to FRED and SFB. We make comparisons with FRED and SFB because these active queue-management schemes are also flow-based. As a reference point, we include results for Droptail queueing to provide a baseline for assessing performance since this is the most widespread queueing policy in effect on the Internet.

We assume that a router knows the bandwidth (L) of the attached outgoing link. N is the number of active flows, i.e., flows that have had at least 1 packet go through the router within a certain *window* of time. Since active flows not experiencing repeated timeouts send several packets per RTT, we use $window = 1sec$, which results in near-perfect estimates in our simulations. We leave more fine tuning of the window parameter for future work. In our experiments, we chose MSS to be 1 KB. We fix the value of c at 1.31, since we use a "random dropping, acknowledgment per packet" model [15].

Since IDMaps aims to provide delay estimates within a factor of two, we simulate the effect of IDMaps by estimating

each RTT to be a uniformly random number between one-half and two times the actual RTT (as inferred from the topology). In the absence of real-world data for IDMaps RTT estimates, we believe that this model will give us an idea of the worst-case performance of GREEN-IDMaps, assuming that IDMaps will usually provide better estimates than this.

We used *ns-2* [17] to evaluate the performance of GREEN over a network with the topology shown in Fig. 1. We try to simulate an organizational topology with low latencies to the "left" of the bottleneck edge router (GREEN). We simulate connections of varying RTTs for the links on the "right" and vary their latencies linearly from $1ms$ to $200ms$. This results in RTTs varying linearly from $72ms$ to $470ms$. N Sources and N sinks are connected to the routers over $10Mbps$ links. We varied the number of flows, N, from 100 to 400. The bottleneck link has a bandwidth of 155 Mbps and a delay of 30 ms. We set the buffer size in the GREEN router to be 600 packets, which is roughly the bandwidth-delay product of the link (581.25 packets), making it consistent with current practices.

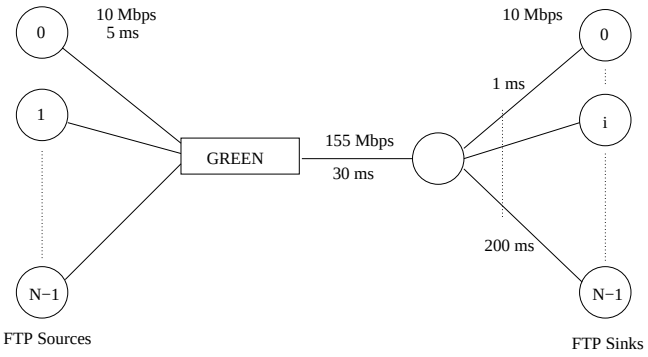


Figure 1: Network Topology

While the background traffic of each experiment varies, we start several FTP connections from the leftmost nodes to the rightmost nodes and run the simulation for $180sec$ and evaluate the fairness between the FTP flows. We run the simulations for that long because unfairness arises from the long-term effects of aggressive shorter RTT flows grabbing bandwidth from longer RTT flows. We only measure the fairness between long-term FTP flows; low-bandwidth and short-lived flows are not of concern because they are usually unaffected by long-term traffic. GREEN-Embedded and GREEN-IDMaps were implemented at the gateway, which is the bottleneck router in our simulation. All of the metrics presented in this section — link utilization, fairness, packet loss, queue size — are measured at this gateway.

6.1 Fairness

As mentioned in Section 2, GREEN attempts to regulate all TCP flows to their fair share of the outgoing link bandwidth. We use Jain's Fairness Index [3] to assess GREEN's ability to maintain equal bandwidths between TCP flows. We briefly describe how the fairness index is calculated, and then present our results.

Jain's Fairness Index: Given a set of throughputs $(x_1, x_2, \ldots, x_n)$, the fairness index is calculated as follows:

$$f(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

The fairness index always lies between 0 and 1. Hence, a higher fairness index indicates better fairness between flows. The fairness index is 1 when all the throughputs are equal. When all the throughputs are not equal, the fairness index drops below 1.

6.2 Link Utilization

Here we compare GREEN's performance with other schemes in terms of overall link utilization. At the end of each simulation, the overall link utilization is calculated as follows:

$$utilization = \frac{byte\_departures_t}{bandwidth \times t}$$

The numerator equals the total number of bytes delivered by the link during the interval of $t\ sec$, and the denominator equals the total possible bytes that could have left the link in the same interval.

6.3 Packet Loss

As we will see, (e.g., Fig. 2(d)), the packet-loss percentage is roughly the same for all flows and stays well below 2%. Equation (1) provides good estimates when p is the order of a few percent [15]. Since the overall packet loss stays below a few percent in our simulations, both GREEN-Embedded and GREEN-IDMaps are able to limit the rates of flows to their fair share of bandwidth.

6.4 Queue Size

Queue sizes are sampled at $20ms$ intervals. The average and standard deviation were calculated at the end of the simulation.

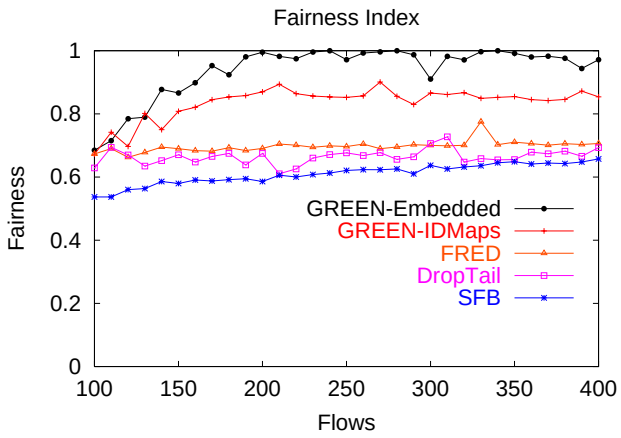
7. EXPERIMENTS AND EVALUATION

In this section we present three sets of experiments. In all experiments, we vary the number of foreground FTP flows from 100 to 400. The first set of experiments does not include any background traffic. The second set of experiments includes background *Pareto* traffic in order to study GREEN's behavior with short-lived connections such as web traffic (so called "web-mice"); these experiments will show that GREEN is resilient to such background traffic. In the third set of experiments, we study GREEN's behavior in the presence of low-bandwidth flows (e.g., flows that are bottlenecked at other routers, or are artificially limited by TCP's flow control window). Again we show that GREEN is able to compensate for such behavior and results in good link utilization. In particular, the last two experiments in this set will show how we have overcome a severe limitation of the older version of GREEN [6]—the inability to maintain high link utilization in the presence of low-bandwidth and short-lived flows.

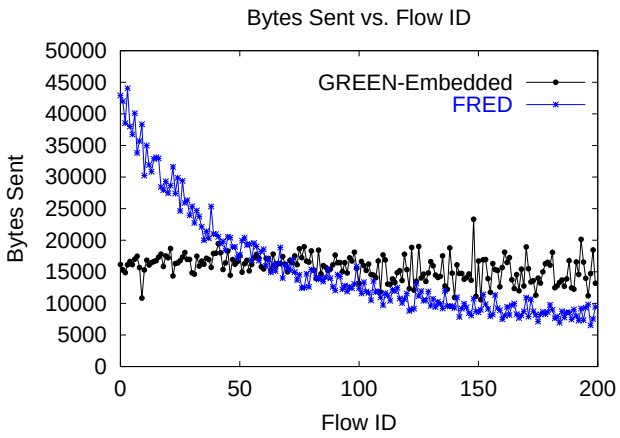
7.1 Experiment 1

7.1.1 Fairness

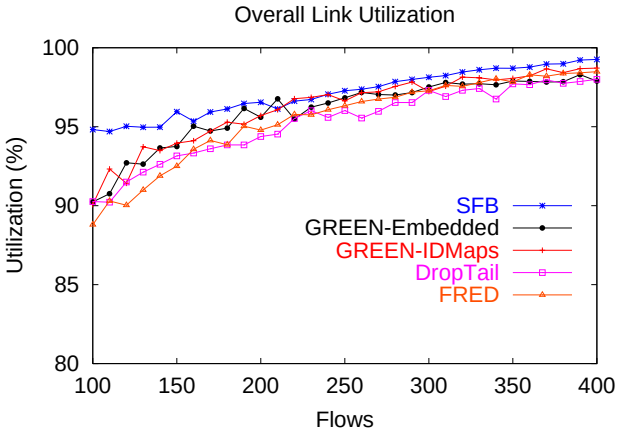
As shown in Fig. 2(a), GREEN-Embedded provides significantly higher bandwidth fairness than the other queue management schemes. The curve for Droptail shows us the fairness we would expect at most gateways in the Internet today.



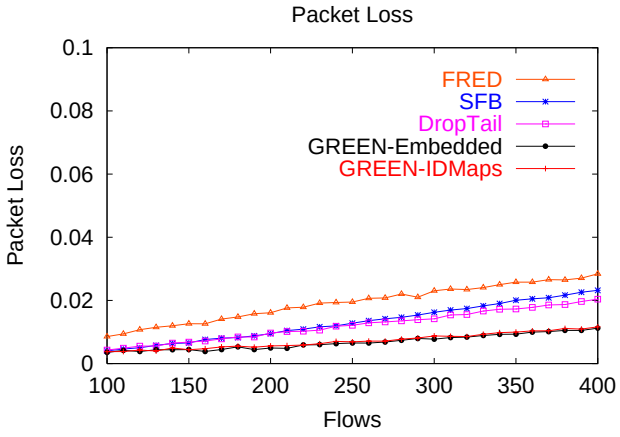
(a) Jain's Fairness Index vs. Number of Flows



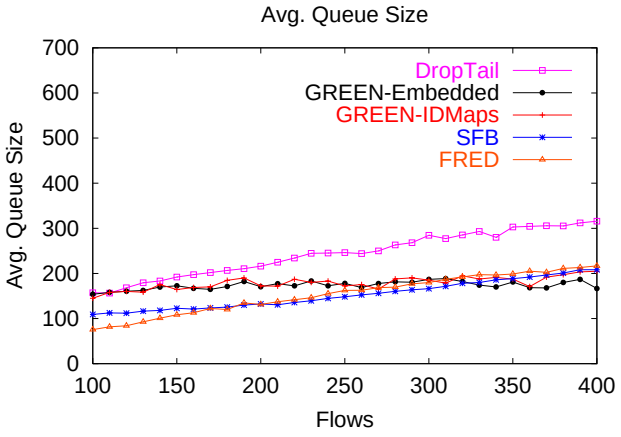
(b) Bytes Sent vs. Flow ID, RTT increases as Flow ID increases



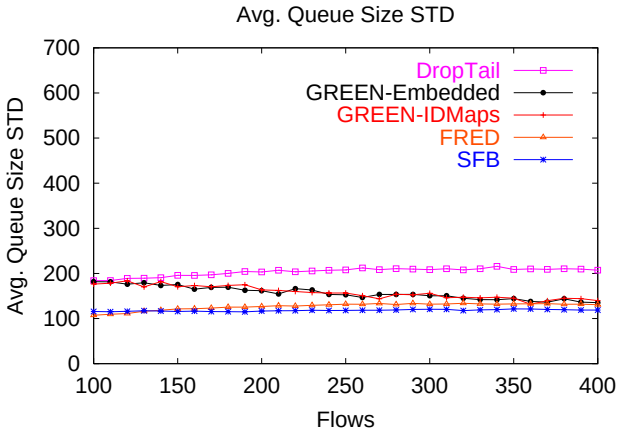
(c) Overall Link Utilization vs. Number of Flows



(d) Overall Packet-Loss vs. Number of Flows

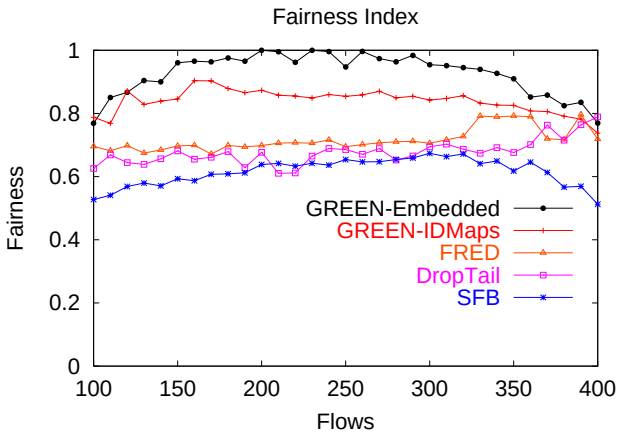


(e) Avg. Queue Size vs. Number of Flows

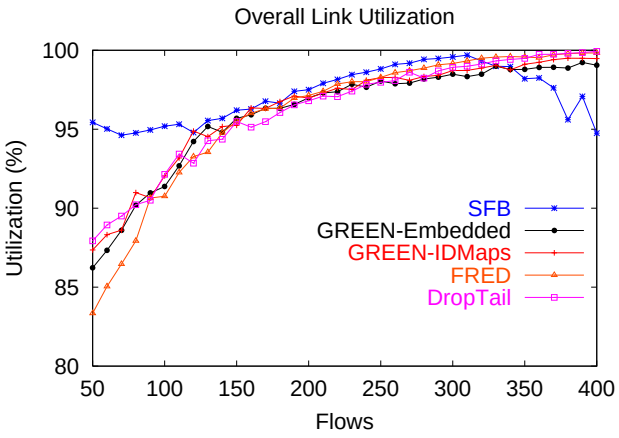


(f) STD in Queue Size vs. Number of Flows

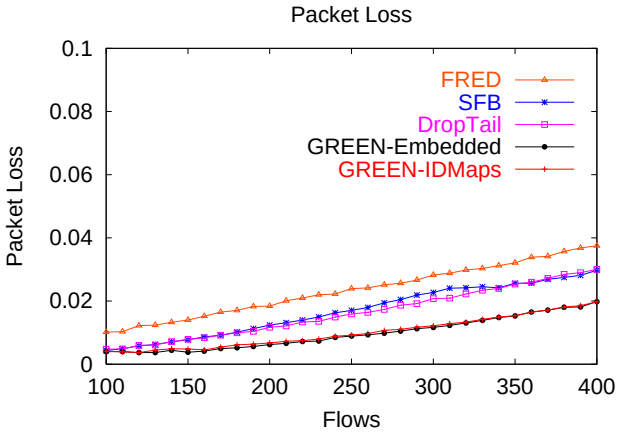
Figure 2: Experiment 1: Results for FTP Flows, No Background Traffic



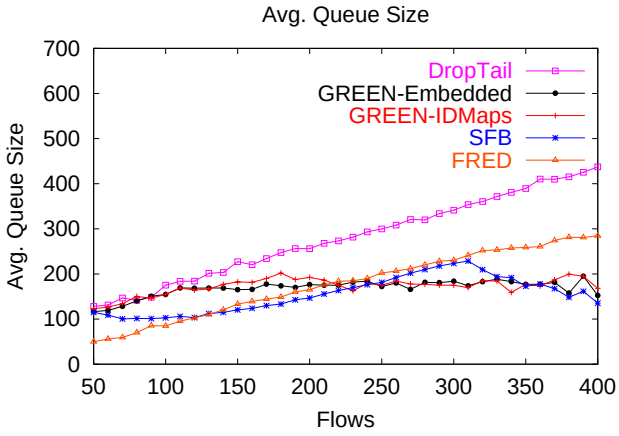
(a) Jain's Fairness Index vs. Number of Flows



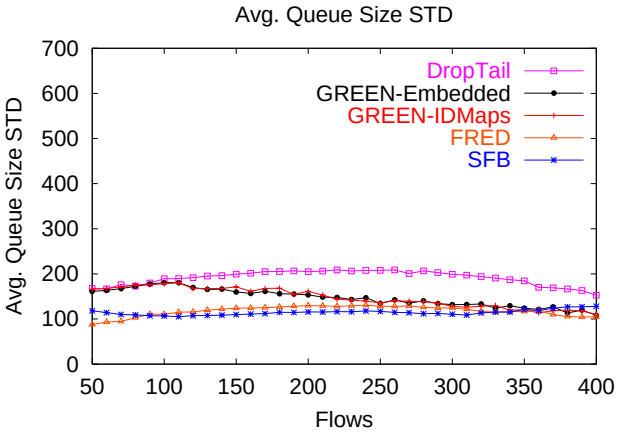
(b) Overall Link Utilization vs. Number of Flows



(c) Overall Packet-Loss vs. Number of Flows

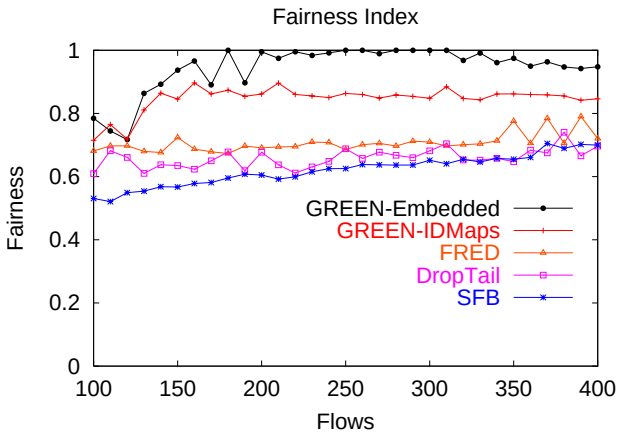


(d) Avg. Queue Size vs. Number of Flows

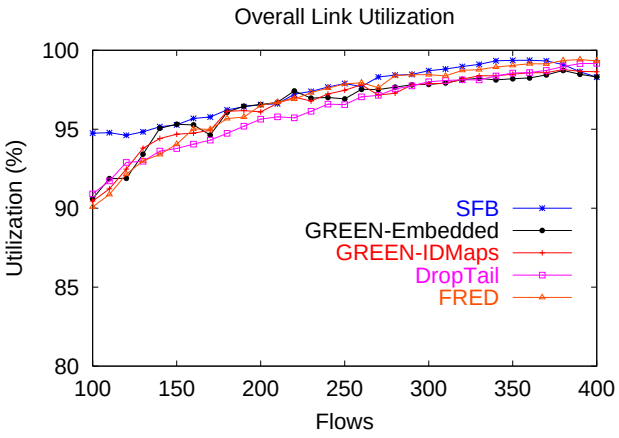


(e) STD in Queue Size vs. Number of Flows

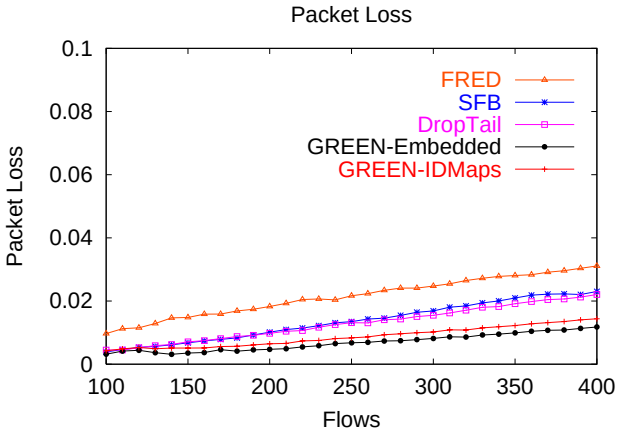
Figure 3: Experiment 2: Results for FTP Flows, Pareto Background Traffic



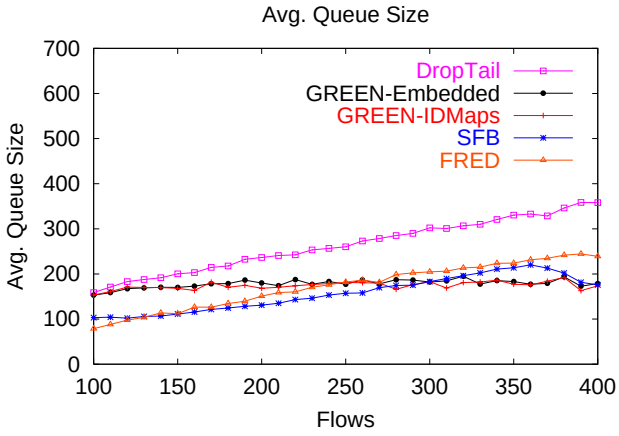
(a) Jain's Fairness Index vs. Number of Flows



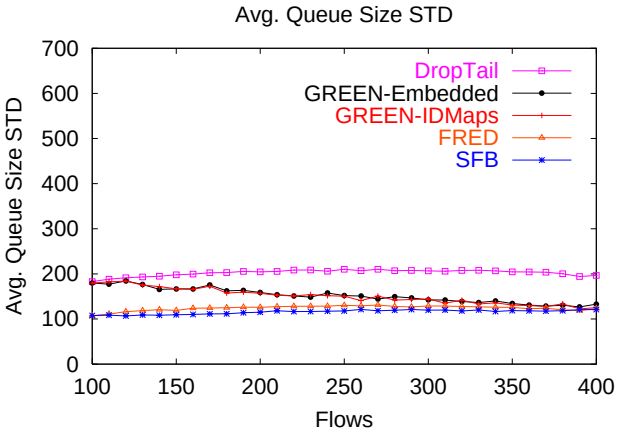
(b) Overall Link Utilization vs. Number of Flows



(c) Overall Packet-Loss vs. Number of Flows



(d) Avg. Queue Size vs. Number of Flows



(e) STD in Queue Size vs. Number of Flows

Figure 4: Experiment 3: Results for FTP Flows, Low-bandwidth FTP Background Traffic

FRED is able to outperform Droptail and SFB because it queues at least two packets of a flow before marking a packet from that flow. This provides much better fairness as long as each flow maintains one to two outstanding packets at the gateway. SFB exhibits poor fairness because it is sensitive to varying RTTs between flows, and breaks down under a large number of connections with varying RTTs [5].

Figure 2(b) best summarizes the benefit of GREEN's approach. At the end of our simulation with 200 FTP flows, we plot the number of packets sent by each flow. We can see TCP's inherent bias against larger-RTT flows in the curve for FRED (Droptail and SFB exhibited similar behavior, and we leave this out for clarity). However, we observe that GREEN has corrected TCP's bias, and all flows achieve roughly the same average bandwidths at the end of the simulation.

GREEN-IDMaps is able to achieve bandwidth fairness much better than FRED, SFB, and Droptail despite the rough RTT estimates. While the fairness provided by GREEN-IDMaps is not as good as that provided by GREEN-Embedded, we can see that a solution based on IDMaps is indeed practical and can be deployed in the Internet. The reason for this is that despite its inaccuracy in RTT estimation, GREEN-IDMaps is able to distinguish between longer and shorter RTT flows.

Fig. 2(c) shows that GREEN-Embedded achieves comparable link utilization to the other queue management schemes. However, SFB is able to provide better utilization when there is lesser contention for the link's bandwidth (fewer than 150 flows). This is because SFB is able to maintain more stable queue sizes with lower variance, resulting is more utilization of the link.

GREEN-IDMaps provides superior fairness compared to Droptail, FRED, and SFB. Even though IDMaps estimates can be off by a factor of two, GREEN is still able to distinguish longer RTT flows from shorter RTT flows. This makes the version of GREEN based on IDMaps attractive since it has high link utilization as well as a high fairness index, and does not require any modification to the TCP sender. Hence it is the simplest solution for deployment, assuming a global service for RTT estimates.

In Figures 2(e) and 2(f), we see that as the number of flows increases, the average queue size for Droptail increases more dramatically than other schemes. In contrast, GREEN-Embedded, GREEN-IDMaps, FRED, and SFB are able to keep the average queue sizes low. In our simulations we observe that the average queue sizes for SFB and FRED gradually increase with the number of flows. While GREEN also displays this behavior for smaller number of flows, the average queue sizes stabilize and is less dependent on the number of flows. We also note that GREEN's queue sizes display Droptail like behavior for a small number of flows. This is because there is less contention for the bandwidth, and hence GREEN's drop probabilities are very low. As the number of flows increases (and hence the contention for bandwidth), GREEN is able to more effectively control queue sizes. We also note that FRED and SFB have the best performance in terms of queue-size variance (resulting in less "jitter" for flows). GREEN is also able to keep variance low, and is comparable to SFB and FRED for a higher number of flows.

FRED keeps queue sizes low by marking packets beyond a certain threshold and limiting the amount of buffer-space for each flow. SFB does so by increasing drop rates when there is packet-loss and reducing drop rates when the link is underutilized. Hence, FRED and SFB attempt to dynamically converge to the correct "operating point" for low queue sizes. GREEN-Embedded achieves its operating point by calculating drop probabilities for each flow, based on their fair share of bandwidth. By ensuring that the aggregate bandwidth of the flows is equal to the available bandwidth at the link, there is no sustained buildup in queue length.

Lastly, we note that has the lowest packet-loss since it able to apply the "correct" drop probabilities to each flow, resulting in fewer overflows at the queue.

In summary, we see that GREEN performs extremely well even when accurate RTT information is not available. This is because slightly inaccurate estimates are still effective in differentiating between flows of longer RTTs and shorter RTTs. The use of IDMaps, which is still a proposed service that has not yet been deployed, motivates the study of other practical RTT estimation techniques.

7.2 Experiment 2

In this experiment, we introduce background *Pareto* flows to simulate the presence of web based traffic—intermittent, short lived connections. We pair each FTP flow with a Pareto flow with the same link characteristics. This is essentially two copies of the topology shown in Fig. 1, sharing the same bottleneck link and GREEN router. One copy runs the FTP flows discussed in the previous experiments, and the other copy runs Pareto flows with the following parameters: $packetsize = 1000bytes$, $bursttime = 2sec$, $idletime = 5sec$, and $rate = 160Kbps$.

As the authors of [6] note, (their version of) GREEN will overestimate the number of contending flows and this will result in the underutilization of the link. In fact, without our approach, we would expect GREEN's utilization to be as low as 50%! This was one of the serious drawbacks of GREEN. In this set of simulations we show how $\gamma(t)$ is able to correct for this behavior and result in high utilization (comparable to other schemes) of the link, and hence rendering GREEN usable in practice.

7.2.1 Results

In Fig. 3(a) we can see that GREEN-Embedded and GREEN-IDMaps exhibit higher fairness than FRED, Droptail and SFB. The reasons are as explained in Section 6.1. However as packet loss approaches about 2%, GREEN's performance degrades. This is because the model [15] that GREEN uses becomes less accurate when there is high contention at the link. The remaining results are similar to that of Experiment 1. We notice comparable link utilization to other schemes (Figure 3(b)), low queue sizes (Figure 3(d)), and low variance in queue size (Figure 3(e)).

7.3 Experiment 3

In this experiment we pair each FTP flow with a corresponding low bandwidth FTP flow, which shares the same link characteristics of the FTP flow, except that the links are set to 56 Kbps to simulate dial-up connections. Again, the older version of GREEN suffers from the same drawbacks as mentioned in Experiment 2, and here we show how $\gamma(t)$ is able to correct for this behavior for low bandwidth FTP flows.

7.3.1 Results

In Fig. 4(a) we can see that GREEN-Embedded and GREEN-IDMaps exhibit higher fairness than FRED, Droptail and SFB. The reasons are as explained in Section 6.1. However as packet loss stays well below 2%, GREEN's performance does not degrade as in Experiment 2. This is because the background traffic exhibits lesser variance, and GREEN is able to share the remaining bandwidth more effectively with the foreground FTP flows. Hence Experiment 2 (with Pareto background traffic) shows how the variability in traffic does affect GREEN's performance for higher traffic loads.

Again, we notice comparable link utilization to other schemes (Figure 3(b)), low queue sizes (Figure 3(d)), and low variance in queue size (Figure 3(e)).

8. STATE REQUIREMENTS

GREEN-Embedded does not maintain any per-flow state. This savings, however, comes at the cost of modifying the TCP senders that embed their RTT estimates within the packet's header. We believe this is feasible in practice because GREEN is more appropriate as an edge router. Organizations can easily deploy modified TCP implementations within their network for better performance.

The basic operation of GREEN-IDMaps does not require per-flow state information. N and MSS can be easily estimated. Since we propose that GREEN-IDMaps routers operate as IDMaps tracers, GREEN-IDMaps will maintain state proportional to the number of tracers deployed in the Internet. The amount of state used depends on how the tracers are connected through virtual links. This is discussed in more detail in [9]. The advantage of this method is that TCP senders do not need to be modified, but this comes at the cost of deploying an IDMaps-like service, which does not exist at this point.

FRED keeps per-flow state information for flows that have packets buffered at the link. SFB does not maintain per-flow state information, but instead, employs a Bloom filter [2] to hash flows into L levels of N bins. Each bin maintains queue occupancy statistics for flows that map into that bin and a corresponding drop probability p_m. Hence, SFB's state requirement is $O(N * L)$. A discussion on the selection of L and N is discussed in [5].

Note that the computational requirements for GREEN are not that demanding. For each packet, GREEN calculates the corresponding drop probability for that flow. We compare this with SFB, which computes L hashes for each packet, updates statistics for that flow, and then calculates the drop probability for that flow.

9. CONCLUSION

Previous work on GREEN [6] suffered from the drawback of serious under-utilization at the link in the presence of short-lived and low-bandwidth flows. Furthermore, the authors assumed that RTT information was available at the router and presented results based on this assumption.

In this paper, we propose a practical implementation of GREEN based on two approaches for estimating a flow's RTT—embedded RTTs and IDMaps, and discuss their tradeoffs. We correct the utilization limitation by introducing the $\gamma(t)$ parameter, which we show is able to effectively utilize the link-bandwidth in the presence of short-lived and low bandwidth flows. Our practical version of GREEN is able to provide superior fairness to other schemes, while maintaining comparable link utilization, low average queue size, and low queue size variance. Furthermore, unlike other stateless approaches that attempt to approximate Fair Queueing, our approach does not require coordination between routers, or significant modifications to the TCP implementation.

10. ACKNOWLEDGMENTS

The authors wish to thank Prof. Wu-chang Feng from the OGI School of Science & Engineering at the Oregon Health and Science University for his insightful comments and feedback on his Blue and Stochastic Fair Blue (SFB) active-queue management schemes as well as his contributions on the framework and viability of the GREEN approach.

We also would like to thank the following people: Sunil Thulasidasan for his initial implementation of GREEN and the following people for their helpful comments: Guanghui He, Vikas Kawadia, Xue Liu, Prasad Naldurg, Charuhas Pandit, Geetanjali Sampemane, Honghai Zhang, Rong Zheng.

11. REFERENCES

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active Queue Management. In *IEEE Network*, June 2001.

[2] B. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communications of the ACM*, 13(7), July 1970.

[3] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin. BLUE: A New Class of Active Queue Management Algorithms. Technical report, April 1999.

[5] W. Feng, D. Kandlur, D. Saha, and K. Shin. Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness. In *Proc. of IEEE INFOCOM 2001*, April 2001.

[6] W. Feng, A. Kapadia, and S. Thulasidasan. GREEN: Proactive Queue Management over a Best-Effort Network. In *Proc. of IEEE Globecom 2002*, November 2002.

[7] S. Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks Part1: One-way Traffic. *ACM Computer Communication Review*, 21(5):30–47, Oct. 1991.

[8] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[9] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, October 2001.

[10] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *IEEE INFOCOM*, 2001.

[11] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *Proc. of ACM SIGCOMM 2002*.

[12] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *ACM SIGCOMM*, 2001.

[13] T. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[14] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proc. of ACM SIGCOMM 1997*, September 1997.

[15] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communication Review*, 27(3), July 1997.

[16] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of INFOCOM*, 2002.

[17] ns-2. Network Simulator. http://www.isi.edu/nsnam/ns.

[18] R. Pan, B. Prabhakar, and K. Psounis. CHOKe A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *Proceedings of INFOCOM*, 2000.

[19] T.Henderson, E. Sahouria, S. McCanne, and R. Katz. On Improving the Fairness of TCP Congestion Avoidance. In *Proc. of IEEE Globecom*, November 1998.