Differentiated Services with Lottery Queueing

Joseph Eggleston Sugih Jamin*

Electrical Engineering and Computer Science Department
University of Michigan
Ann Arbor, MI 48109-2122
{*jeggle,jamin*}*@eecs.umich.edu*

Abstract. This work investigates the benefits and drawbacks of using a lottery to schedule and drop packets within a router. We find that a lottery can provide many distinct levels of service without requiring that per-flow state be maintained in the routers. We also investigate the re-ordering within a flow that can occur if a lottery is used to pick packets to forward without regard to their flow order.

1 Introduction

The current Internet infrastructure provides the same level of service to all packets, namely Best Effort service. This type of service has proven to be adequate for many applications. However, given the heterogenous nature of traffic on the Internet and that network bandwidth on the Internet is a limited resource, a means for providing different levels of service to different traffic would be beneficial. Higher levels of service could be used to meet the QoS needs of certain applications (e.g., streaming audio and video), or to simply satisfy the demands of users who are willing to pay for improved service.

Before different levels of service can be provided, the network needs a way to determine the level of service packets should receive. There are two prominent methods for making this distinction. Either the router maintains enough state to identify the level of service a packet should receive based on the flow it belongs to, or packets carry the state needed to identify their service requirements. The Integrated Services (IntServ) framework [1] is a system based on the first method, while the Differentiated Services (DiffServ) model [2], [3] proposes the second.

Because it requires routers to maintain per-flow state, IntServ is generally considered less scalable than DiffServ. DiffServ preserves the stateless nature of core routers thus guarding their scalability. It does this by distinguishing between core routers and boundary routers (also known as edge routers). Boundary routers (ingress, egress, or possibly both) perform traffic shaping and policing of flows leaving the core routers to forward packets based on state carried in packet headers. This state identifies the aggregate traffic class to which that packet belongs. Core routers use this state to determine

* This project is funded in part by DARPA/ITO grant F30602-97-1-0228 from the Information Survivability program. Sugih Jamin is further supported by the NSF CAREER Award ANI-9734145, the Presidential Early Career Award for Scientists and Engineers (PECASE) 1999, the Alfred P. Sloan Research Fellowship 2001, and by equipment grants from Sun Microsystems Inc. and Compaq Corp.

how the packet should be treated, that is, the state determines the Per-Hop Behavior (PHB) the packet receives. PHB groups typically define a few traffic classes and the associated forwarding behavior those classes receive. Examples of PHBs are Assured Forwarding [4] and Expedited Forwarding [5]. Assured Forwarding prescribes four traffic classes with up to three drop precedences per class. Differentiation is achieved by assigning each class a portion of the available bandwidth and buffer space, and through use of the drop precedence. Expedited Forwarding provides one high priority class with guarantees of a certain amount of bandwidth available for packets in that class.

This work investigates the aptitude of Lottery Queueing for providing Differentiated Services. Lottery Queueing is based on the CPU scheduling mechanism presented in [6].[1] In Lottery Queueing, each packet carries a bid value set at the sender. Routers use the bid value to distinguish between packets waiting in their forwarding queues. The bid value can be used both to determine the next packet to forward and, when necessary, to choose a packet to drop. In this way, Lottery Queueing can provide differentiation in either latency, or drop rate, or both. Ideally, there will be a large range of bid values available which will allow for many levels of service. Because bid values are carried in the packet headers, it preserves the stateless nature of routers implementing it.

In this paper we show that not only can Lottery Queueing provide differentiation in both latency and drop rate between packets carrying different bid values, it does so in a manner proportional to the bid values. That is, Lottery Scheduling can provide many levels of service dependent on the number of unique bid values.

Using a taxonomy similar to that in [2], the service provided by Lottery Scheduling is best described as Probabilistically Relatively Quantified. That is, in the aggregate, packets with higher bids will receive better service than packets with lower bids. However, because it is probabilistic, individual packets with a higher bid may experience worse service than a packet with a lower bid.

Probabilistically Relatively Quantified service may be adequate for some applications. For example, assuming that Best Effort traffic is assigned a bid value of one, someone simply desiring faster than Best Effort service could use a bid value greater than one. However, some applications may have more specific service requirements. If the sender knows that a certain packet carries "important" data, it could use a higher bid value for that packet. Dynamically changing the bid value could also allow a flow to maintain a certain level of service despite changing network conditions. Because it is the receiver that knows when service requirements are not being met, a feedback mechanism from the receiver to the sender could be used. This feedback mechanism could take the form of explicit requests for a bid change, or could be inferred at the sender using Acks. The key attribute of Lottery Queueing that allows user feedback to the network for meeting QoS needs is its ability to allow fine grain changes in service dependent only on the number of bid values available.

Because allocating bandwidth is a zero-sum game, there must be some penalty that prevents users from trying to grab the largest share by always using the maximum bid value. The most obvious such penalty is charging users more for using higher bid values. Differentiated Services assumes that a customer will negotiate a contract with a

[1] The authors of [6] uses the term Lottery Scheduling. We also use that term, but for a specific aspect of Lottery Queueing.

network provider that specifies the guarantees provided by the network provider and the restrictions on the traffic the customer can send into the provider's network. This agreement is referred to as a Service Level Agreement (SLA). SLAs will be recursively formed between neighbor networks from the sender to the receiver so that some type of end-to-end guarantee exists. One way that Lottery Queueing could fit into this framework is for domains to agree to forward a certain amount of traffic at a set bid value. For example, there could be an agreement to forward 10 Mbps of bid 10 traffic. A more interesting use of Lottery Queueing, the one we focus on, allows senders to set and dynamically change the bid value of their flows. For this purpose, an SLA could include pricing agreements for packets carrying different bid values. If restrictions are not placed on the amount of traffic carrying each bid value, the result will be a market for available bandwidth where a larger share of the bandwidth goes to those who pay more for using higher bid values.

In Sect. 2 we describe the mechanisms behind Lottery Queueing. Section 3 relates how we implemented the test system. Section 4 presents experimental results. It includes experiments on drop rate, queueing latency, and packet re-ordering. Finally, Sect. 5 concludes the paper.

1.1 Related Work

Lottery Queueing does not provide fair share scheduling in the sense of Fair Queueing [7]. If all flows use the same bid value, the flow that sends the most will get the largest share of bandwidth. However, if flows are charged for the packets they send, Lottery Queueing could be considered to achieve a type of fairness where paying more gives you better service.

Core-Stateless Fair Queueing (CSFQ) [8] and Core-Jitter Virtual Clock (CJVC) [9] both endeavor to provide Fair Queueing while maintaining the stateless nature of the network core. CSFQ uses edge routers to mark packets with their flow's arrival rate and uses that information in the core to provide Fair Queueing at core routers. It does not provide for allocating different portions of the bandwidth to different flows. CJVC goes beyond Fair Queueing and provides guaranteed service for flows. However, it assumes a reservation based admission control policy. While Lottery Scheduling does not provide guaranteed service, its more relaxed service model can provide many levels of differentiation with less strict admission control such as Measurement-Based Admission Control [10] or no admission control.

RED [11] provides a means for congestion avoidance. RED routers track the average queue length. When the average queue length is less than a set minimum value, no packets are dropped. When the average queue length is greater than a set maximum value, all packets are dropped. Average queue lengths between the minimum and maximum values cause packets to be dropped with a certain probability. RIO [12] extends the RED system to provide two service levels by distinguishing between in and out packets. Out packets are dropped earlier and with a higher probability than in packets. It would be possible to extend Lottery Scheduling to provide a congestion control mechanism combined with service differentiation as in RIO. We already use bid values to differentiate between packets when dropping packets due to queue overflow. This could

be extended to include Early-Drop with lower bid packets having a higher probability of being dropped early. We leave exploration of this possibility as future work.

2 Packet Scheduling

We describe the queueing systems we have investigated. We focus on Lottery Queueing for its potential to provide many levels of service with proportional sharing. For comparison purposes, we also examine Deterministic Queueing which uses the bid value to forward based on static priority. A queueing system consists of the scheduling discipline, which determines the next packet to be served, and a dropping policy, which determines the packet to drop when the queue is full.

2.1 Scheduling Disciplines

We study two scheduling disciplines that can differentiate based on a bid value: Lottery Scheduling and Deterministic (Static Priority) Scheduling. For comparison purposes, we also include traditional FIFO (first-in-first-out) scheduling in our study.

Lottery Scheduling is a probabilistic scheduling method. The next packet to be served is chosen by holding a lottery with the probability of an individual packet being served being proportional to its bid value:

$$\Pr[\text{packet } k \text{ is served}] = \frac{B_k}{\sum_j B_j}, \tag{1}$$

where B_i is the bid value of packet i; j represents each packet in queue. With lottery scheduling, packets with higher bids have a greater chance of being served next. Therefore, during times of congestion higher bid packets should typically experience lower queueing delay than packets with lower bids. However, since low bid packets nevertheless will have a chance of being served, they won't be starved. Even if high bid packets keep arriving low bid packets will still receive a share of the bandwidth proportional to their bid value.

One side effect of Lottery Scheduling is that packets in a flow have a higher probability of arriving out of order at the receiver. When a flow has more than one packet in a router's queue, each packet has equal probability of being forwarded next (assuming the flow has not changed the bid value carried by its packets). We explore this aspect of Lottery Scheduling in more detail in Sect. 4.4.

An alternative form of Lottery Scheduling that does not suffer from this problem would maintain the order of a flow's packets by always sending in FIFO order within a flow. This does not require a router to maintain state for every flow passing through it, it only needs to keep track of flows that currently have packets in the queue. If there are two packets from the same flow in the queue, the router would ensure that the first packt to enter the queue is forwarded first.

Deterministic Scheduling always forwards the packet with the highest bid. If the queue contains multiple packets with the same highest bid value they are served in

FIFO order. As long as a flow does not change its bid value, Deterministic Scheduling will not re-order queued packets belonging to a flow.[2] The key difference between Lottery Scheduling and Deterministic Scheduling is that Lottery Scheduling provides proportional scheduling. That is, with Lottery Scheduling, all flows will receive service at a level proportional to their bid value; Deterministic Scheduling allows starvation of lower bid packets if higher bid packets keep arriving.

2.2 Dropping Policies

We consider three dropping policies: Lottery Drop, Deterministic Drop, and the standard Drop Tail policies.

The mechanism behind Lottery Drop is analogous to that of Lottery Scheduling. When a new packet arrives at a full queue it is placed in the queue and a lottery is held to determine which packet to drop. For Lottery Drop the probability of an individual packet being dropped is proportional to the inverse of its bid value.

$$\Pr[\text{packet } k \text{ is dropped}] = \frac{1/B_k}{\sum_j 1/B_j} \tag{2}$$

Hence higher bid packets are less likely to be dropped than lower bid packets.

Deterministic Drop selects the lowest bid packet in the queue to drop. If there is more than one packet with the same lowest bid, the latest to arrive is dropped.

2.3 Interactions of Scheduling and Dropping Policies

As stated above, a queueing mechanism is defined by its scheduling discipline and dropping policy. When combined into a single queueing mechanism, the scheduling discipline and dropping policy are no longer independent. For example, focusing on the time spent by a single packet in the queue, the number of packets served prior to this packet influences the chances of this packet being dropped. Likewise, the number of other packets dropped influences the amount of time the packet must wait before it sees service. Hence, it is important to compare complete queueing mechanisms rather than just the forwarding or dropping policies. The following are the combinations of scheduling disciplines and dropping policies we use in this study:

1. FIFO scheduling with Drop Tail (FSDT). This is the traditional router queueing mechanism.
2. Lottery Scheduling with Drop Tail (LSDT). Lottery Scheduling provides lower latencies for higher bid packets. Drop Tail guarantees that once a packet enters the queue it will eventually be served.
3. Lottery Scheduling with Lottery Drop (LSLD). This combination tends to favor higher bid packets in both forwarding and dropping. A packet that has entered the queue is not guaranteed to be forwarded.

[2] However, packets can still arrive out of order at the receiver due to network topological or routing changes.

4. FIFO Scheduling with Lottery Drop (FSLD). This combination favors higher bid packets only when there is enough congestion to cause the queue to overflow.
5. Deterministic Scheduling with Deterministic Drop (DSDD). This combination always forwards the highest bid packet and drops the lowest bid packet.

3 Implementation

In this section we describe our implementation of the studied packet queueing systems.[3] This implementation has not been optimized; there are certainly more efficient ways of implementing Lottery Queueing. The implementation was intended only to allow us to study the behavior of Lottery Queueing. We have implemented our new queueing mechanisms in the FreeBSD 3.2 operating system kernel. The main modifications are changes to the general Ethernet driver and the Ethernet interface device driver [13]. While two queues exist for each interface, an input queue and an output queue, we modify only the output queue since the CPU on our machines processes packets fast enough to avoid queueing in the input queue. There are only two modifications to the kernel queue data structure: the addition of a variable to hold the sum of all bids in the queue and a variable to hold the sum of all inverse bids. These variables facilitate holding lotteries for scheduling and dropping packets as described below. Whenever a packet is added to or removed from the queue these variables are updated.

Packet Enqueueing. Enqueueing is done by the general Ethernet driver. This is also where packets are dropped if the queue is full. The standard Drop Tail implementation simply adds the packet to the end of the queue. If the queue is full, the packet is dropped and the memory used by the packet is freed.

When using Lottery Drop, an incoming packet is always added to the end of the queue. If the queue is full a lottery is then held by pseudo-randomly picking a number between zero and the sum of all inverse bids in the queue using the inverse bid sum stored in the queue structure. The queue is traversed, keeping a running sum of the inverse bid values as packets are passed, until the running sum exceeds the randomly chosen value, in which case the corresponding packet is dropped.

For Deterministic Drop, incoming packets are inserted in bid order (high to low). When the queue becomes full, the packet at the end of the queue is dropped.

Packet Dequeueing. Dequeueing is done by the Ethernet interface device driver. The standard FIFO scheduling queue simply picks the packet from the front of the queue for sending.

The Lottery Scheduling mechanism works in a manner similar to Lottery Drop. To pick a packet for sending, a number between zero and the sum of all bids in the queue is chosen pseudo-randomly. The queue is traversed, keeping a running sum of bid values as packets are passed, until the running sum exceeds the randomly chosen value, in which case the corresponding packet is forwarded.

Deterministic Scheduling forwards the packet at the front of the sorted queue.

[3] Our reference implementation is available at http://irl.eecs.umich.edu/jamin/papers/marx/lottery.tgz.

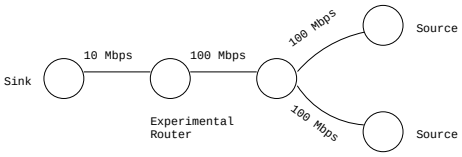


Fig. 1. Testbed setup. The congestion point is between the Experimental Router and the Sink.

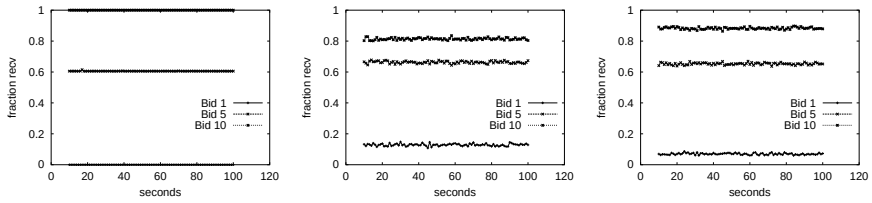
Bid Placement. For Lottery Queueing, we require a means of carrying the bid value inside the packet. For the purposes of the experiments in this paper we place the bid value in the IPv4 type-of-service (TOS) field. At this time, we have made no attempt to fit the bid field into the the Differentiated Services fields of IPv4 and IPv6 defined in [14]. We note that Lottery Queueing will take advantage of as many bits as are made available for bids by providing more levels of service.

4 Experimental Results

We used the testbed shown in Fig. 1 in our experiments. During our tests the Source nodes send data to the Sink node. The route includes a congestion point after the Experimental Router which is running one of the experimental queueing systems. We collect data from the arriving packets at the Sink node. All packets are sent using the unreliable, connectionless User Datagram Protocol (UDP). We now show the efficacy of lottery queueing in differentiating packets with varying bids. The differentiation takes the form of both packet loss and queue waiting time. To get an idea of the basic behavior of our queueing systems, we first explore differentiation in drop rate and queue latency using CBR traffic (Sections 4.1 and 4.2). Section 4.3 describes results using more realistic traffic models. We explore the possibility of packet re-ordering due to Lottery Scheduling in Sect. 4.4.

4.1 Drop Rate

We first explore the ability of Lottery Drop and Deterministic Drop to differentiate between packets of varying bids. For this experiment, we use three source processes, each generating 374-byte packets (headers included) at a constant rate of 2000 packets per second. Through experimentation we determined that the maximum throughput of the network through the congestion point is about 3220 packets per second; each process sends at 62% of link capacity. Each process marks its packets with a different bid value: 1, 5, and 10. Packets are also marked with sequence numbers, allowing the Sink node to track which packets arrive and which are dropped at the Experimental Router. We run each experiment for 120 seconds using each of the queueing schemes described above. We allow the system to warm up and stabilize for 10 seconds before any data is collected. Figure 2 shows the percent of packets belonging to each flow that makes it to the Sink node under each of the queueing mechanism. We do not present the results



(a) DSDD (b) FSLD (c) LSLD

Fig. 2. Percent of packets received by each flow under three queueing mechanisms.

of FSDT and LSDT since the drop-tail policy they use does not differentiate packets by their bid values, so each flow gets an equal share of the available bandwidth.

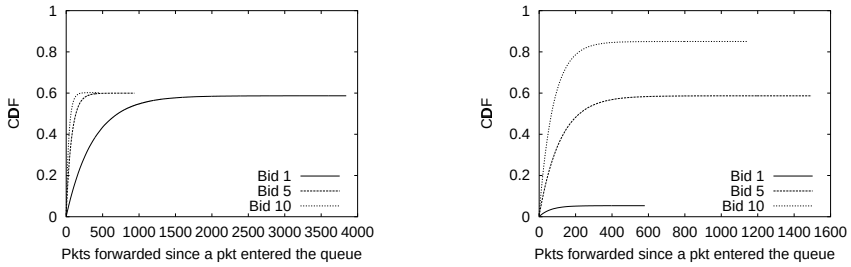
The DSDD results in Fig. 2a show that the high bid flow receives as much bandwidth as it needs. The bid 5 flow takes the remaining available bandwidth, leaving the low bid flow completely starved.

FSLD (Fig. 2b) and LSLD (Fig. 2c) give each flow a share of the available bandwidth proportional to their bid values. The differences between the FSLD and the LSLD results show that the scheduling algorithm used affects packet loss rate. LSLD shows more differentiation between the varying bid values. This is because Lottery Scheduling tends to forward the high bid packets before the low bid packets, causing the low bid packets to go through more lottery drops and increasing their chance of being dropped.

4.2 Queueing Delay

We next examine the amount of time packets with different bid values must wait in the queue before they are forwarded under the various queueing mechanisms. The Experimental Router keeps a count of the number of packets forwarded from a queue since the start of the experiment. By stamping each packet with the value of this counter as it enters and leaves the queue, we obtain a measure of the queueing delay for each packet. The queueing delay is thus expressed in terms of the number of other packets forwarded between the time a packet enters and leaves the queue. The Sink node collects the queueing delay information from all packets. It is important to note that we obtain no queueing delay data from dropped packets. To account for dropped packets in presenting the queueing delay data, we use the total number of packets sent, as opposed to the total number of packets received, to compute the cumulative distribution function (CDF) of queueing delays.

The data presented here is collected from the same experiments used to present the drop rate data in the previous section. The queue length at the Experimental Router was set to 160 packets. Packets forwarded through the FSDT queue have to wait a full queue of 159 packets 93% of the time, and they never wait less than 154 packets. This indicates that most packets that are not dropped filled the queue to capacity. Occasionally, the three sending processes become synchronized such that several packets arrive nearly simultaneously and are dropped, giving the queue a chance to drain slightly. This information is mostly interesting for comparison with the other queueing methods.



(a) LSDT (b) LSLD

Fig. 3. Queueing delay under two queueing mechanisms. The CDFs do not reach 1 reflecting the amount of packets dropped.

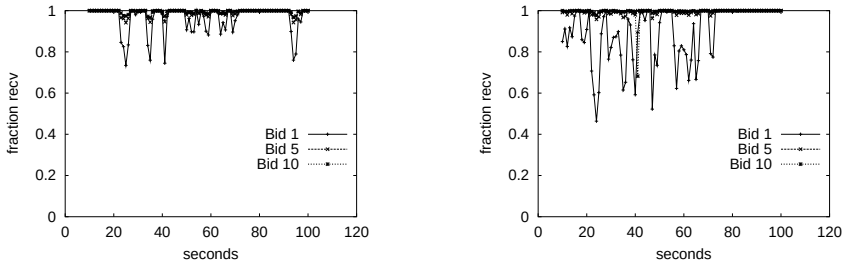
DSDD always forwards the highest bid packet in the queue first. In this experiment high bid packets see immediate service 98.7% of the time. Bid 5 packets wait an average of 420 packets. Since the combination of the high and medium flows completely saturates the congested link, bid 1 packets are all dropped, leaving the queue full of bid 5 packets. Not only do the bid 5 packets have to wait for the full queue to drain before they see service, service is interrupted every time a high bid packet arrives at the queue, making the wait longer.

We present the CDFs of queueing delay under LSDT (Fig. 3a) and LSLD (Fig. 3b). The CDFs do not go up to 1, reflecting the amount of packets dropped. While the drop policy of LSDT does not differentiate packets by their bid values in making drop decisions, the use of Lottery Scheduling does take the bid values into account in forwarding packets. This is reflected in the difference in queueing delay CDFs of the various bid values. The LSLD CDFs demonstrate an interesting and non-intuitive interaction between the scheduling and dropping policy. The tail for the Bid 1 CDF is much shorter than the tails for the higher bid CDFs suggesting that the low bid packets see faster service. Because Lottery Drop is being used, the low bid packets are dropped preferentially over the high bid packets. If a low bid packet is not forwarded quickly from the queue, it is likely it will be dropped. Therefore, the sink only counts the low bid packets that happen to be selected quickly for forwarding. The maximum queueing delay in the distributions is much higher than the queue size of 160 packets because packets are served at random. Packets with the highest bid value must contend with others of the same bid value for service, hence they also can see queueing delay longer than the queue size.

We do not present the queueing delay CDFs for FSLD because the FIFO scheduling discipline does not differentiate service by bid value, hence all packets see the same queueing delay distribution.

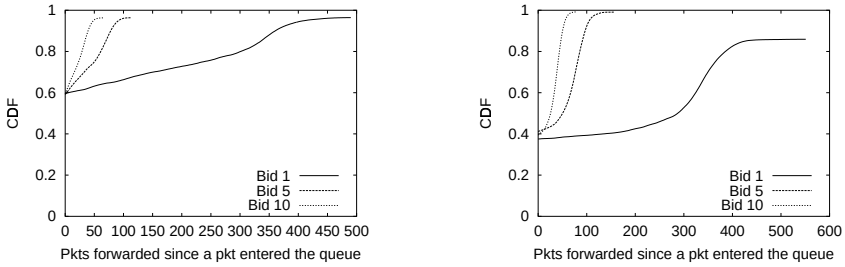
4.3 Long-Range Dependent Traffic

All the flows in the experiment described above transmitted at constant bit rate. Researchers in network traffic characterization have observed long-range dependency in aggregate network traffic [15]. To study the effectiveness of our queueing mechanisms



(a) FSLD (b) LSLD

Fig. 4. Fraction of packets received by each flow under two queueing mechanisms for Pareto on/off sources.



(a) LSDT (b) LSLD

Fig. 5. Queueing delay under two queueing mechanisms for Pareto On/Off sources. The CDFs do not reach 1 reflecting the amount of packets dropped.

on long-range dependent traffic, we conduct a similar experiment on sources generating on-off traffic with Pareto distributed on and off times. Aggregate traffic from such sources has been shown to exhibit long-range dependency [16].

Figures 4a and 4b show that in the face of long-range dependent (LRD) traffic, while higher bid flows continue to receive preferential treatment under FSLD, and an exaggerated preferential treatment under LSLD, lower bid flows do not suffer as much as in the previous case. The high variance of long-range dependent traffic allows lower bidding traffic to continue to be served at network switches, albeit with a longer delay. Hence, when network traffic is very bursty, lower bidding traffic experiences longer queueing delay but not higher loss rate. This effect can be seen by comparing Fig. 3a against Fig. 5a. Note that the CDFs of all bid values are higher in the latter graph where the traffic is generated by Pareto on/off sources; however, the percentage of lower bid packets staying longer in the queue is much higher. This effect is even more pronounced under the LSLD queueing mechanism, as can be seen by comparing Fig. 2c against Fig. 4b for drop rates of CBR and Pareto on/off sources, respectively, and the corresponding queueing delays shown in Figures 3b and 5b, respectively.

4.4 Out-of-Order Packets

In this section we analyze the proclivity of Lottery Scheduling to re-order packets within the network. As shown in (1), Lottery Scheduling chooses the next packet to forward without any knowledge of the flow that packet belongs to. This means that each packet a flow has in the queue has an equal opportunity of being the next packet forwarded (assuming each of those packets carry the same bid value). To assist in understanding how serious this effect will be, we have analyzed a simplified model of our queueing schemes that include Lottery Scheduling as their scheduling mechanism. This model gives some insight into what will affect the possibility of re-ordering and the extent to which re-ordering will take place if Lottery Scheduling is used. The most significant result found using this model is that the main parameters governing the degree of re-ordering a flow experiences at a router are the percentage of incoming traffic to the router belonging to the flow and the bid value used by the flow. Increasing the percentage of incoming traffic increases the degree of re-ordering, while increasing bid value decreases the degree of re-ordering.

Model Description. In our simplified model all sources are CBR sources and the sum of those sources is still CBR. That is, there is no burstiness in the traffic.[4] With these CBR sources, if the total incoming traffic to a queue is less than the outgoing link speed there will never be any packets waiting in the queue to be sent, and so there is never any chance of packets being reordered. Therefore, the only interesting case is when the total incoming traffic is greater than the outgoing link speed. In this case, the queue will always remain full since packets are arriving faster than they can be forwarded. Since the queue length is constant (always full) the total number of incoming packets must equal the total number of outgoing packets, where a packet is outgoing when it is either forwarded or dropped. Since the queue is in equilibrium, it follows that the number of packets that a single flow has in the queue will remain constant.

The following system of equations helps describe the behavior of this model using the LSLD queueing scheme:

$$\frac{n_i \cdot b_i}{\sum_{\text{all flows}} n_j \cdot b_j} \cdot f_{forw} + \frac{\frac{n_j}{b_j}}{\sum_{\text{all flows}} \frac{n_j}{b_j}} \cdot f_{drop} = f_i \tag{3}$$

$$\sum_{\text{all flows}} n_i = qlen \tag{4}$$

n_x: the number of packets flow x has in the queue.
b_x: the bid value that flow x marks its packets with.
f_x: the fraction of incoming packets belonging to flow x.
f_{forw}: the fraction of total incoming packets forwarded.
f_{drop}: the fraction of total incoming packets dropped. This is $1 - f_{forw}$.
$qlen$: the length of the queue in packets.

[4] While the model assumes CBR traffic, we believe the results give insight into more realistic traffic as well. For example, packets from ON/OFF traffic can only be re-ordered while the source is "ON", and while the source is "ON" it is sending at a constant bit rate.

Note that (3) actually represents a whole system of equations, one for each unique flow using the queue. Equation (4) provides a constraint that guarantees the solution to the system is one where the queue is full. Equation (3) represents how flow i is treated in the queue. The right side of the equation gives the percentage of the incoming traffic belonging to flow i. The left side of the equation represents the fact that each packet of flow i must either be forwarded or dropped. The left half of the left side is the portion of forwarded packets belonging to flow i. The right half of the left side is the portion of dropped packets that belong to flow i.

The equations representing LSDT are similar. In this case, because Drop Tail is used, the fraction of dropped packets belonging to a flow is equal to that flow's fraction of the total incoming traffic. Therefore, a flow's fraction of the total incoming traffic is equal to the flow's fraction of packets entering the queue after dropping. This, in turn, is equal to the flow's portion of all forwarded packets. From this we get the following equation replacing (3) in the system above:

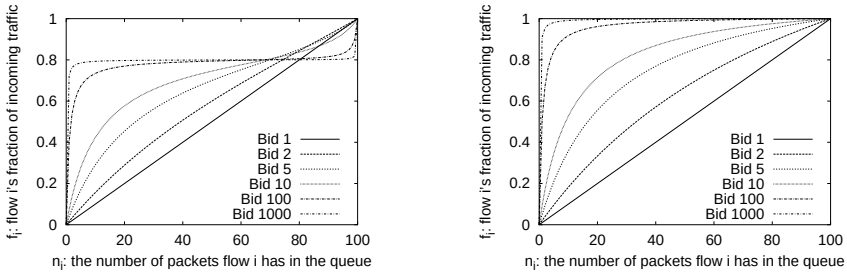
$$\frac{n_i \cdot b_i}{\sum_{\text{all flows}} n_j \cdot b_j} = f_i \tag{5}$$

Using these systems of equations we can predict the number of packets each flow will have in the queue when the respective queueing scheme is used. This is useful because the probability that the next packet belonging to a flow forwarded from a router is out-of-order is determined by the number of packets that flow has in the queue as seen in the following equation.

$$\text{Pr(Out of Order)} = 1 - \frac{1}{n_x} \tag{6}$$

Where Pr(Out of Order) is the probability that flow x's next packet sent from the queue is out-of-order. Intuitively, the more packets a flow has in the queue, the greater the chance that the next forwarded packet belonging to that flow will be out-of-order.

Predicted Behavior. Figure 6a is helpful in understanding the behavior predicted by the LSLD system of equations. The x-axis of the graph marks the number of packets a flow has in the queue, n_i; the y-axis is the percentage of total incoming packets belonging to that flow, f_i. Each line represents a different bid value, b_i, for the flow. In all cases, all the incoming traffic not belonging to flow i, $1 - f_i$ percent of the total incoming traffic, has a bid value of 1. f_{forw} and f_{drop} are 0.8 and 0.2 respectively, and $qlen$ is 100. When b_i is 1 the function is a straight line. Intuitively, this makes sense because the background traffic also carries a bid of 1 so LSLD degenerates to randomly picking packets from the queue to forward and drop which implies that the number of packets a flow has in the queue should be directly proportional to the percentage of incoming traffic belonging to that flow. When b_i is 1000 the function is approximately a step-function with the shift in n_i taking place when f_i equals f_{forw}. The step represents the transition from flow i needing less than the outgoing link bandwidth to needing more. Because b_i is so much greater than the bid value of the background traffic, 1000 versus 1, when f_i is less than the outgoing link bandwidth, flow i's packets are forwarded immediately from the queue keeping n_i near 0. When f_i becomes greater than f_{forw}



(a) LSLD (b) LSDT

Fig. 6. The behavior predicted by the LSLD (equation (3) and LSDT (equation 5) models. Each line represents the function relating the fraction of incoming packets belonging to a flow using a certain bid value to the number of packets that flow should expect in the queue at any one time.

flow i's packets start to compete with each other for the outgoing link bandwidth. This leaves the queue full of flow i's packets since the background traffic packets have a much greater chance of being dropped.

Figure 6b uses the same parameters as Fig. 6a to show the behavior of a LSDT system. The figure shows that when using LSDT a high bid flow does not take over the queue until f_i approaches 100% of the incoming bandwidth. This is because all flows are dropped in proportion to their send rate regardless of their bid value, so whatever percentage of the incoming bandwidth that flow takes, it will receive the same percentage of the outgoing bandwidth.

Experimental Verification. To convince ourselves of the accuracy of our equations we ran experiments in a setup approximating our model. Fig. 7 shows the results of those experiments. The line shows the calculated value of n_i on the x-axis for different values of f_i on the y-axis. The parameters used are the same as the bid 10 analysis for LSLD as described above. The individual points show experimental results. f_{forw}, f_{drop}, $qlen$, and b_i were kept constant throughout the experiments matching the values used in the calculations. Based on the observed maximum outgoing link bandwidth we could simply calculate the total amount of incoming traffic needed to get the desired f_{forw} and f_{drop}. Each experiment used a different value of f_i for the bid 10 traffic with bid 1 traffic making up the remaining amount of total incoming traffic. For the purposes of this experiment, we had the router keep track of how many packets each flow had in the queue. Each outgoing packet was marked with the number of packets each flow had in the router's queue at that time. The points in the chart show the average number of packets in the queue while the error bars show the standard deviation. As Fig. 7 shows, the calculated values predicted very well what was observed.

Preventing Out-of-Order Packets. In this section we look at the maximum fraction of incoming bandwidth, f_i, a flow can take while keeping the number of packets it has in the queue less than two. By keeping the number of packets in the queue less than two, a flow is guaranteed not to suffer from re-ordering due to Lottery Scheduling. Of

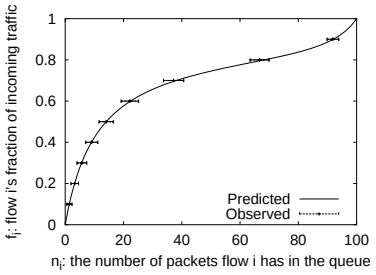


Fig. 7. Calculated vs. experimental packets in queue. The solid line shows the values predicted by (3). The individual points show experimental results.

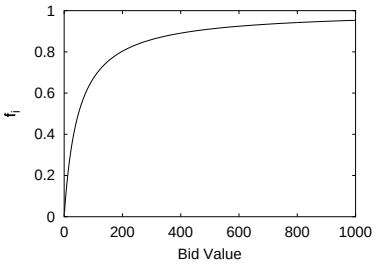


Fig. 8. The maximum fraction of incoming traffic to a router a flow can have with various bid values which keeps the number of packets that flow has in the queue less than 2. This defines the amount of traffic a flow can send through a router without experiencing re-ordering.

course, this f_i is going to vary depending on other flows using the queue and the bid values of all the flows. In this section we look this fraction for different b_i's assuming the remaining traffic carries a bid value of one.

Figure 8 shows the maximum f_i a flow can use while keeping its n_i less than two when the queueing scheme is LSDT or LSLD with f_{forw} equal to 1. (For LSLD with f_{forw} other than 1, the graph will asymptotically approach that value of f_{forw}.) This result is presented for evaluation purposes only. We are not advocating keeping per-flow state in core routers for the purpose of allowing flows to track their f_i's.

5 Conclusion

In this paper, we have examined the possibility of providing different levels of service using a lottery to schedule and drop packets. We have shown that both Lottery Scheduling and Lottery Drop have the ability to provide many distinct levels of service. Lottery Drop provides each flow a share of the bandwidth proportional to the bid value that flow is using. Likewise, flows with higher bids see proportionally faster service when Lottery Scheduling is used. Trials with bursty traffic are especially encouraging. They show that Lottery Scheduling can still provide service differentiation when the queue length fluctuates.

The high volume of traffic usually associated with core routers suggests that Lottery Queueing may be best suited for use in the core. Since it preserves the stateless nature of the core, it scales well with the number of flows. In fact, a large number of flows will benefit Lottery Scheduling. The likelihood of packet re-ordering due to Lottery Scheduling decreases as the share of traffic belonging to any one flow decreases. If any one flow is an insignificant portion of the core traffic, the likelihood of re-ordering will be low.

References

1. S. Shenker, R. Braden, and D. Clark, "Itegrated services in the internet architecture: an overview," RFC 1633, Internet Engineering Task Force, June 1994.
2. Y. Bernet, J. Binder, S. Blake, M. Carlson, B. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss, "*A Framework for Differentiated Services*," Internet Draft, draft-ietf-diffserv-framework-02.txt, Feb. 1999.
3. S. Blake, D. Black, M. Carlson, and E. Davies, "An architecture for differentiated services," RFC 2475, Internet Engineering Task Force, December 1998.
4. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding phb group," RFC 2597, Internet Engineering Task Force, June 1999.
5. V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding phb," RFC 2598, Internet Engineering Task Force, June 1999.
6. C.A. Waldspurger and W.E. Weihl, "Lottery scheduling: Flexible proportional-share resource management," *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI) '94*, pp. 1–12, 1994.
7. A. Demers, S. Keshav, and S.J. Shenker, "Analysis and simulation of a fair queueing algorithm," *Proc. of ACM SIGCOMM '89*, pp. 1–12, Sept. 1989.
8. I. Stoica, S. Shenker, and H. Zhang, "Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks," *Proc. of ACM SIGCOMM '98*, pp. 118–130, Sept. 1998.
9. I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," *Proc. of ACM SIGCOMM '99*, pp. 81–94, Aug. 1999.
10. S. Jamin, P.B. Danzig, S.J. Shenker, and L. Zhang, "A measurement-based admission control algorithm for integrated services packet networks (extended version)," *ACM/IEEE Transactions on Networking*, vol. 5, no. 1, pp. 56–70, Feb. 1997, Available from `http://netweb.usc.edu/jamin/admctl/ton96.ps.Z`.
11. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *ACM/IEEE Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
12. D. Clark. and W. Fang, "Explicit allocation of best effort packet delivery service," *ACM/IEEE Transactions on Networking*, vol. 6, no. 4, pp. 362–373, Aug. 1998.
13. G. Wright and W. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*, Addison-Wesley, 1995.
14. K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers," RFC 2474, Internet Engineering Task Force, December 1998.
15. W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *ACM/IEEE Transactions on Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.
16. W. Willinger, M.S. Taqqu, R. Sherman, and D.V. Wilson, "Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level," *Proc. of ACM SIGCOMM '95*, pp. 100–113, Aug. 1995.