Chardonnay - Achieving Fair Bandwidth Allocation with Priority Dropping Based on Round Trip Time *

Mark Claypool, Bob Kinicki, Choong-Soo Lee
claypool@cs.wpi.edu, rek@cs.wpi.edu, clee01@cs.wpi.edu
Worcester Polytechnic Institute
Computer Science Department
100 Institute Road
Worcester, MA 01609

ABSTRACT

Current congestion control approaches that attempt to provide fair bandwidth allocation among competing flows primarily consider only data rate when making decisions on which packets to drop. However, responsive flows with high round trip times (RTTs) can still receive significantly less bandwidth than responsive flows with low round trip times. This paper proposes a congestion control scheme called "Chardonnay" that addresses router unfairness in handling flows with significantly different RTTs. Using a best-case estimate of a flow's RTT provided in each packet by the flow source or by an edge router, Chardonnay computes a stabilized average RTT. The average RTT is then compared with the RTT of each incoming packet, dynamically adjusting the drop probability so as to protect the bandwidth of flows with high RTTs while curtailing the bandwidth of flows with low RTTs. We present simulation results and analysis that demonstrate that Chardonnay provides better fairness than other rate-based congestion control strategies over a wide-range of traffic conditions. The improved fairness of Chardonnay comes close to the fairness of Fair Queuing without requiring per flow state information at the router.

1. INTRODUCTION

The Internet relies upon cooperation between TCP hosts and subnet routers to adjust source data rates in the presence of network congestion along the path of the TCP flow. Currently, drop-tail queue management is the primary queue mechanism used in Internet routers to indicate congestion to edge hosts. While drop-tail schemes are easy to implement and fit within the best-effort nature of the Internet, these routers distribute packet drops arbitrarily among competing flows.

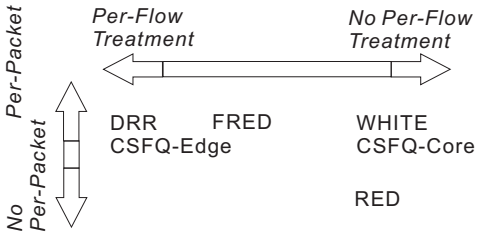

Figure 1: Classification of AQMs

RED [6], the best known Active Queue Management (AQM) approach, uses randomization to distribute packet drops among flows in a manner proportional to their perceived share of the capacity on a bottlenecked link. While RED drops more packets from high bandwidth flows, the same packet drop rate is applied to all flows, regardless of the actual bandwidth used.

Many researchers have shown that end-to-end congestion control can be improved when bandwidth is allocated more fairly among flows [1] [12]. Furthermore, congestion increases due to handling packets that never reach their destination may be the largest unresolved form of congestion collapse in the Internet today [5]. In times of heavy congestion, dropping an equal number of packets among all flows regardless of round trip time (RTT) has the potential to increase the network capacity wasted on undelivered packets that have come a long way only to be dropped by a router before reaching their final destination. This inherent bias against flows with high round trip times decreases overall goodput [1].

For TCP-friendly flows, a flow's round trip time (as well as packet size and drop rate) is directly responsible for determining a flow's data rate [3] [7]. With TCP's congestion control algorithms, a flow's throughput varies inversely with RTT. Thus, especially for flows with low round trip times, small RTT differences can dramatically effect the number of packets dropped for a flow. Approaches that are designed for

*The term Chardonnay comes from a play on the acronym RED. Considering "red" as a type of wine, there is a family of "white wine" active queue management techniques, of which Chardonnay is one.

[1] As in [5], we define the goodput of a flow as the packet delivery rate at the receiver, excluding duplicate packets.

increased fairness over RED [9] [14] do not consider round trip time in making packet dropping decisions and are unfair with respect to heterogeneous flows with wide variances in round trip times.

There have been numerous approaches to achieving per flow fair bandwidth allocation at congested routers. As noted in [11], there is a continuum of possible per-flow treatments, from complete per flow treatment such as in Fair Queuing, to a complete absence of per-flow treatment such as in drop-tail and RED. Additionally, for queuing mechanisms without per-flow treatment, there is a continuum of possible per-packet treatments, from no per-packet treatments such as in RED to complete per packet treatments from CSFQ core routers. Figure 1 depicts the space of possible flow and packet treatment policies, with the approximate placements of policies evaluated in this paper.

Random Early Detection (RED) [6] keeps no per flow state information. Packets are dropped probabilistically based on the long-term average queue size and fixed indicators of congestion (thresholds). RED uses randomization to drop arriving packets to avoid biases against bursty traffic and roughly drop packets in proportion to the flows data rate at the router. However, flows with high RTTs and small window sizes are bursty, and this burstiness causes high variability in the perceived data rate of these flows as seen by RED routers.

At the other extreme, Deficit Round Robin (DRR) [13], a variant of Fair Queuing (FQ) [1], keeps extensive information on every flow. DRR routers send packets approximately in the order a router would send them if packets could be sent one bit at a time. While DRR and other FQ variants achieve good fairness among flows, the per-flow state information required and overhead needed to manage priority queues is expensive. Moreover, these schemes do not scale well with increased number of flows. This study uses DRR as the best case scenario for achieving fairness among heterogeneous flows; namely the goal is to seek fairness comparable to DRR without DRR per-flow costs.

Flow Random Early Drop (FRED) [9] uses per-flow preferential dropping to achieve fairer allocation of bandwidth among flows. FRED builds per-flow state at the router by examining those packets that are currently in the queue. The packet drop rate for a flow is determined by the number of packets the flow has in the queue, and is not directly influenced by the flow's data rate nor round trip time. We evaluate the effectiveness of FRED as a less expensive means of attempting per-flow fairness.

In Core Stateless Fair Queuing (CSFQ) [14], edge routers classify flows based on their current sending rate and forward these data rates as labels to core routers. Using these labels, core routers keep a running estimate of the fair share capacity of a flow on an out-going link. The core router drops packets in a manner aimed at giving each flow its fair share of the link throughput. However, such preferential dropping based on data rate alone is not sufficient to achieve fairness. Since the response function for TCP-friendly flows is based on the RTT, dropping packets equally between two flows with the same data rate but different round trip times will result in a higher long-term data rate for the flow with the lower round trip time. While it has been shown that CSFQ achieves fairness for flows with the same RTT, we demonstrate that it is ineffective in achieving fairness among flows with heterogeneous round trip times.

This paper presents a new approach to fairness that takes into account a flows round trip time in determining a router's responsiveness to congestion avoidance. The primary goal of our work is to achieve fairness [2] among responsive flows with heterogeneous RTTs without negatively impacting overall router performance (i.e., goodput, delay and drop rate).

The framework for our approach, Chardonnay, is the same *edge* and *core* architecture presented in CSFQ [14] wherein core routers drop packets using *hints* sent in packet labels by edge routers. In practice, the edge can be an ingress router or an end host and the hint can consist of the estimated data rate, as in CSFQ, the estimated window size, a delay tolerance, or any other flow attribute. In Chardonnay, the hint is an edge estimate of the best-case round trip time.

Using round trip time hints, the Chardonnay core router computes an average round trip time for all packets arriving at the router. When congestion is indicated by the RED thresholds, packets are dropped based on their round trip time in relation to the overall average round trip time. This mechanism preferentially drops packets with lower than average round trip times and favors packets with higher than average round trip times. This protects fragile flows with high RTTs while inhibiting robust flows with low round trip times from grabbing an unfair share of link bandwidth.

Through NS-2 [10] simulations, Chardonnay's effectiveness is demonstrated under a wide range of scenarios. These scenarios include mixes of flows with round trip times varying from 20 ms to 400 ms, scenarios with a disproportionately large numbers of flows in different round trip time clusters, equal clusters of flows in different round trip time clusters, and scenarios with drastic changes in the round trip times of active flows.

Our simulation results show that the Chardonnay strategy provides far superior fairness among flows than RED for all scenarios tested; achieves fairer link capacity allocation among flows than CSFQ and FRED under all scenarios; provides far better fairness than CSFQ and FRED in many scenarios; yields DRR equivalent fairness for most scenarios; and approximates DRR equivalent fairness for the other scenarios tested. These Chardonnay improvements in fairness are accomplished while providing performance similar to RED with respect to drop rate, goodput and throughput.

The rest of this paper is organized as follows: Section 2 focuses on the details and derivation of the Chardonnay mechanisms; Section 3 describes the set of simulation experiments run to evaluate Chardonnay under a wide-range of conditions and includes analysis of the simulation results and detailed comparisons of the performance of Chardon-

[2] We focus on *min-max fairness*, since it is easy to interpret locally and makes no assumptions about behaviors elsewhere in the network. For completeness *Jain's fairness index* [8] is also reported for all experiments.

nay with DRR, CSFQ, RED and FRED. Section 4 summarizes our findings and considers further extensions and future work.

2. CHARDONNAY

In this section, we present the Chardonnay algorithm, summarized in Figure 2. Each packet contains a round trip time (RTT) label, as described in Section 2.1. For each incoming packet, the RTT label is used to update the RTT average kept by the router, as described in Section 2.2. If there is extreme congestion, indicated by the queue average being above max_{th}, the packet is dropped. If there is congestion, as indicated by the queue average being between min_{th} and max_{th}, the drop probability for the packet is computed based on the queue parameters, the RTT label, and the RTT average, as described in Section 2.3. If the packet is not dropped, it is enqueued in a normal first-in, first-out fashion.

```
on receiving packet p
if (p.RTT > 0) then
    updateAvg(R_average, R_formula, p.RTT)
if (q_avg >= max_th) then
    dropPacket(p, 1)
else
    if (avg >= min_th) then
        d = calcDropProbability(q_avg, min_th, max_th,
                                p.RTT, R_formula))
        dropPacket(p, d)
    else
        enQueuePacket(p)
```

Figure 2: Chardonnay Algorithm

2.1 Average Round Trip Time at the Edge

As in [14], we assume an architecture where edge routers on the ingress of a network cloud label a packet with additional information, called an *edge hint* so that core routers on the interior of the network cloud can make packet drop decisions efficiently. In our case, the edge hint is given by the sending host using TCP-Reno and it is the lowest RTT recorded, as computed using baseRTT computation from TCP Vegas code.

Based on discussion in [15], there are from 4 to 17 bits available that can be used to carry edge hint information. We store the RTT in the IP packet using a 16-bit integer, but in practice our range of RTTs would only require about 9 bits. Moreover, 9 bits still allows coverage of up to 80% of RTTs typically observed [2]. At a granularity of 10 ms, 9 bits would be sufficient to cover RTT ranges of up to 5 seconds. Further evaluation of the impact on using variable IP bits for edge hints on both fairness and network overhead we leave as future work.

2.2 Round Trip Time at the Router

The average RTT at the router is calculated using an exponential weighted moving average, called $R_{average}$. To adjust quickly to changes in the networks, the weight w_{RTT} is set to 0.1, which is much higher than a typical w_q of 0.002 used as a weight for a RED queue average. The value of 0.1 is necessary to quickly detect changes in the RTT of incoming packets, caused by the the addition of new flows, the termination of old flows or a change in route of some flows. To prevent excessive variation in RTT under steady state, the algorithm uses a stabilized measure of the RTT, called $R_{formula}$, to compute drop probabilities (see Section 2.3). $R_{formula}$ is only changed when $R_{average}$ has significantly moved (defined to be an interval of 25ms centered around $R_{formula}$ in our implementation) for an entire time interval (defined to be 100ms, an approximate RTT in our implementation). If $R_{average}$ has moved from one side of the range to the other, called *crossing over* in Figure 3, we reset the time interval. When it is determined that $R_{average}$ has moved, $R_{forumla}$ is updated to move towards the new $R_{average}$.

```
now = getCurrentTime()
R_average = (1 - w_RTT) R_average + (w_RTT) p.RTT
if (((R_formula-12.5) < R_average AND
    (R_formula+12.5) > R_average)
    OR R_average crossed over) then
        lasttime = now
else
    if ((now - lasttime) > 100ms) then
        lasttime = now
        update R_forumula towards new R_average
```

Figure 3: Algorithm for Computing Round Trip Time at the Router

2.3 Drop Probability Based on Round Trip Time

In order to compute a drop probability based on a packet's RTT relative to the average RTT ($R_{average}$), we start with the TCP response function [7]:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\sqrt{\frac{3p}{8}}p\,(1+32p^2)} \qquad (1)$$

This provides the upper bound on the sending rate of T as a function of the packet size s, steady state loss rate p, round trip time R, retransmission time out t_{RTO}. Although the drop probability in RED is not exactly equivalent to the steady state loss rate p, p will be used to estimate the relationship between the drop probability and the round trip time. Combining the three terms involving p above using a constant and exponent, we get the simplified equation:

$$T = \frac{s}{Rcp^a} \qquad (2)$$

Consider two flows with throughputs T_1 and T_2 and round trip times R_1 and R_2, respectively. In order to achieve fair bandwidth allocation, T_1 and T_2 should be equal. This leads to the following derivation:

$$T_1 = T_2$$

$$\frac{s}{R_1 cp_1^a} = \frac{s}{R_2 cp_2^a}$$

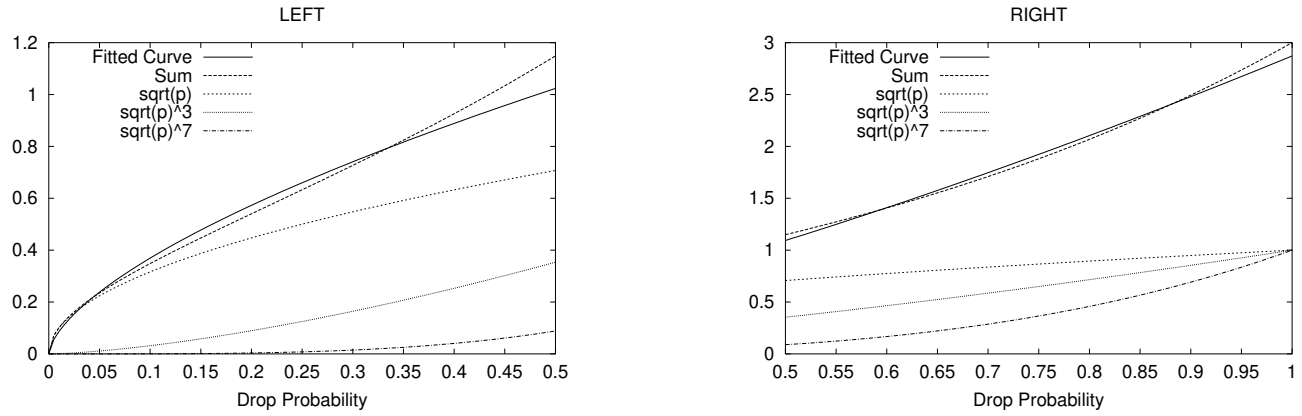

Figure 4: Contribution of p-Terms vs. Drop Probability.

$$p_2^a = p_1^a \left(\frac{R_1}{R_2}\right)$$

$$p_2 = p_1 \left(\frac{R_1}{R_2}\right)^{\frac{1}{a}} \quad (3)$$

The exponent a needs to summarize the behavior of the denominator of the original equation. The three terms involving p in equation (2) have exponents of 1/2, 3/2 and 7/2. When p approaches 0, $p^{1/2}$ dominates, but when p approaches 1.0, each p-term contributes nearly equally. We divide the region of p into two sections, 0.0 to 0.5 and 0.5 to 1.0, and calculate a best-fit power curve for the sum of the p terms for each region. Figure 4 depicts each p-term's contribution to the sum, and the best fit curve to the sum. When p is small, Figure 4-*left* shows that the sum of the p-terms follows closely to square root of p as expected, with the estimated exponent a in the fitted curve function being about 0.63. When p approaches 1, Figure 4-*right* shows that each p-term contributes to the sum, with the estimated exponent a in the fitted curve function being about 1.39.

Chardonnay starts with a base drop probability p computed from min_{th}, max_{th}, max_p and q_{avg}, as in traditional RED [6]. For robust flows whose RTTs are smaller than the average RTT, their drop probability, p_{robust}, should be higher than the average drop probability p_{base}. Therefore, the second and the third p terms dominate, leading to the exponent of the curve fit from Figure 4-*right*. This leads to a drop probability of:

$$p_{robust} = p_{base} \left(\frac{R_{formula}}{R_{robust}}\right)^{\frac{1}{1.39}}$$

$$p_{robust} = p_{base} \left(\frac{R_{formula}}{R_{robust}}\right)^{0.71} \quad (4)$$

For fragile flows whose RTTs are higher than the average RTT, their drop probability, $p_{fragile}$, should be lower than p_{base}, leading to the exponent of the of the curve fit from Figure 4-*left*. This leads to a drop probability of:

$$p_{fragile} = p_{base} \left(\frac{R_{formula}}{R_{fragile}}\right)^{\frac{1}{0.63}}$$

$$p_{fragile} = p_{base} \left(\frac{R_{formula}}{R_{fragile}}\right)^{1.58} \quad (5)$$

For the rest of the paper, the exponent used for the drop probability for robust flows will be called α and the exponent used for the drop probability for fragile flows will be called β.

Chardonnay has a basic complexity similar to that of RED. In addition, Chardonnay has four additional variables (α, β $R_{average}$, and $R_{formula}$). For each packet that arrives, Chardonnay must read the roundtrip time label in each packet, and compute $R_{average}$, adjusting $R_{formula}$ as necessary. No per flow state information is required.

3. EXPERIMENTS

In this section, we evaluate Chardonnay by comparing its performance with the four additional algorithms mentioned in the introduction: RED, with no explicit attempt at fairness as the current status quo of (un) fairness; DRR, with packet scheduling to achieve bandwidth fairness as the model of fairness; and CSFQ and FRED as less complex ways of achieving the fairness sought by DRR. We used the NS-2 simulator [10], which provides packet-level implementation of many TCP protocols and many buffer queue management algorithms including DRR and RED. For CSFQ and FRED, we used the code developed in [14].

We implemented Chardonnay by extending the existing RED code with the drop probability algorithms described in Section 2. Then we set up the network topology shown in Figure 7. There are 30 sources, N_0 *through* N_{29} going through a bottleneck router R to destination D. The bottleneck bandwidth is 10 Mbps and the delay 5 ms. The settings for RED, FRED, CSFQ and Chardonnay are in Figure 7. For CSFQ, the averaging constants K (used in estimating the flow rate), K_α (used in estimating the fair rate), and K_c (used in making the decision on whether the link is congested or not)

AQM	Experiment											
	1	2	3	4	5				6			
					A	B	C	D	A	B	C	D
RED	0.785	0.872	0.930	0.982	0.680	0.769	0.654	0.867	0.844	0.885	0.812	0.931
FRED	0.828	0.867	0.788	0.992	0.738	0.823	0.642	0.867	0.883	0.900	0.801	0.930
CSFQ	0.781	0.796	0.860	0.977	0.740	0.647	0.641	0.610	0.915	0.734	0.965	0.986
DRR	0.927	0.975	0.984	0.991	0.880	0.933	0.960	0.970	0.962	0.972	0.985	0.977
Chardonnay	0.982	0.993	0.985	0.991	0.888	0.959	0.903	0.915	0.971	0.976	0.970	0.948

Figure 5: Jain's Fairness Index for All Experiments

AQM	Experiment			
	1	2	3	4
RED	[0.15, 0.79]	[1.47, 5.98]	[0.27, 0.85]	[0.16, 0.40]
FRED	[0.14, 0.69]	[1.85, 4.98]	[0.24, 1.17]	[0.21, 0.38]
CSFQ	[0.14, 0.82]	[1.98, 5.66]	[0.27, 1.04]	[0.07, 0.38]
DRR	[0.20, 0.53]	[2.84, 3.50]	[0.21, 0.42]	[0.22, 0.37]
Chardonnay	[0.25, 0.40]	[3.14, 3.56]	[0.36, 0.53]	[0.28, 0.39]

AQM	Experiment							
	5				6			
	A	B	C	D	A	B	C	D
RED	[1.76, 4.78]	[3.26, 6.48]	[2.61, 7.01]	[3.81, 5.86]	[0.68, 5.87]	[2.25, 7.60]	[1.45, 8.26]	[3.13, 6.57]
FRED	[1.87, 4.60]	[3.41, 6.15]	[2.50, 7.13]	[3.68, 6.21]	[1.01, 5.68]	[2.62, 6.92]	[1.31, 8.28]	[3.14, 6.77]
CSFQ	[2.25, 4.41]	[2.02, 7.82]	[4.07, 5.57]	[4.46, 5.26]	[1.57, 5.87]	[1.36, 8.61]	[1.20, 7.22]	[1.07, 8.89]
DRR	[2.70, 3.68]	[4.66, 5.30]	[4.93, 4.96]	[4.84, 5.10]	[1.84, 4.57]	[4.24, 5.71]	[4.24, 5.63]	[4.60, 5.33]
Chardonnay	[3.09, 3.28]	[4.51, 5.33]	[4.38, 5.26]	[4.72, 5.08]	[2.26, 4.49]	[4.11, 5.77]	[4.16, 5.47]	[3.97, 5.81]

Figure 6: Minimum and Maximum Goodput (Mbps) for All Experiments

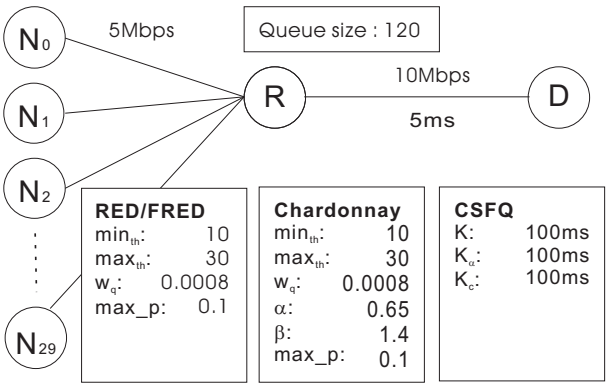

Figure 7: Network Topology and Settings

are set as recommended in [14]. Any setting not listed in the figure is set to the default NS-2 configuration. All flows use TCP-Reno with the RTT modification described in Section 2.2 and a maximum window size of 64 packets.

A set of six experiments were run using this topology and all settings are the same across experiments, except that the latencies between each node N_i and the router R differ, and the active queue management (AQM) technique was one of RED, FRED, CSFQ, DRR or Chardonnay. To numerically compare the fairness, we computed Jain's fairness index [8] for each experiment (Figure 5). Jain's fairness index takes the average goodputs of the flows and computes a normalized number between 0 and 1, where 0 denotes the maximum unfairness and 1 denotes the maximum fairness. We also collected the minimum and maximum goodput values in Figure 6.

In general, Chardonnay achieves reasonable fairness, significantly closer to DRR than RED, and Chardonnay is moderately fairer than CSFQ and FRED. In each experiment, there are a large number of dynamics, some of them rather complex. Due to space limitation, we merely highlight the important points and will put further details in a technical report in the near future.

3.1 Uniformly Distributed Latencies

Experiment 1 considers many sources with various roundtrip latencies. Each source latency is calculated using Equation 6, which introduces a linear increase in latency from one source to the next.

$$latency\,(N_i) = 2\,[(i+1)\,5ms + 5ms] \qquad (6)$$

Therefore, the 30 sources have round trip latencies ranging from 20 ms to 310 ms. The results of experiment 1 were used to tune the α and β parameters for Chardonnay. Although the theoretical values derived in Section 2 were 0.7177 and 1.578 respectively, the actual values used based on the best results from this experiment, as shown in Figure 7, are 0.65 and 1.4.

The experiment simulations were run for 150 seconds and the goodput was averaged over a 120 second period starting

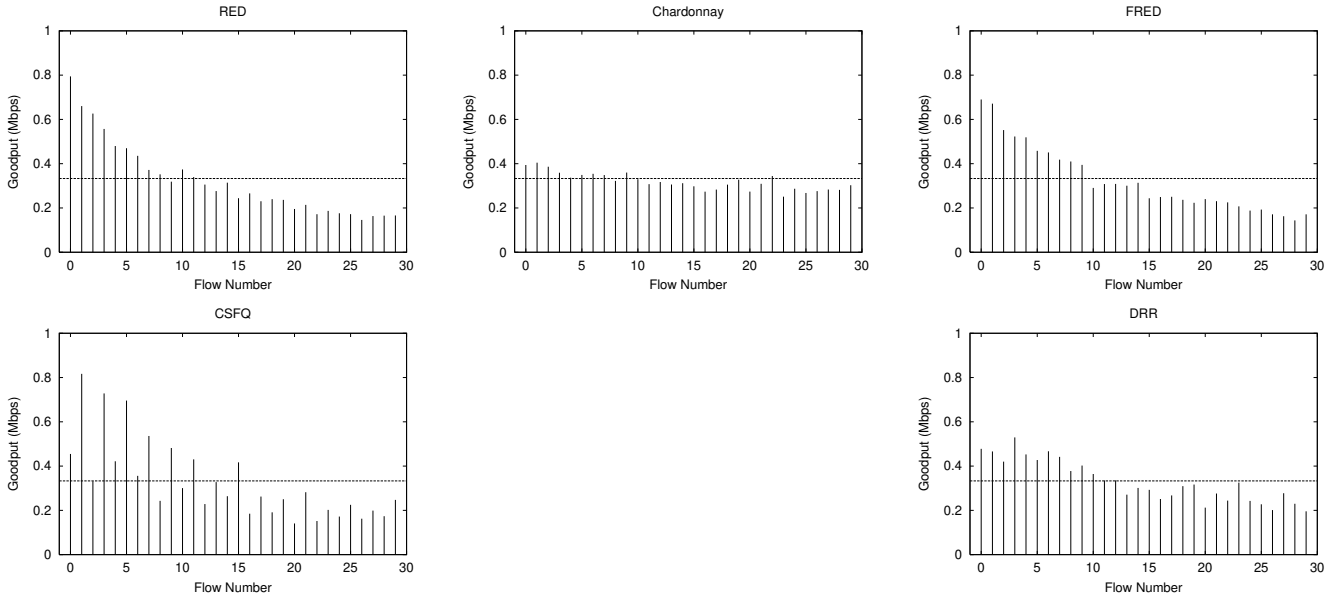

Figure 8: Experiment 1 (Uniformly Distributed Latencies). The 30 flows have roundtrip latencies ranging from 20 ms (left) to 310 ms (right).

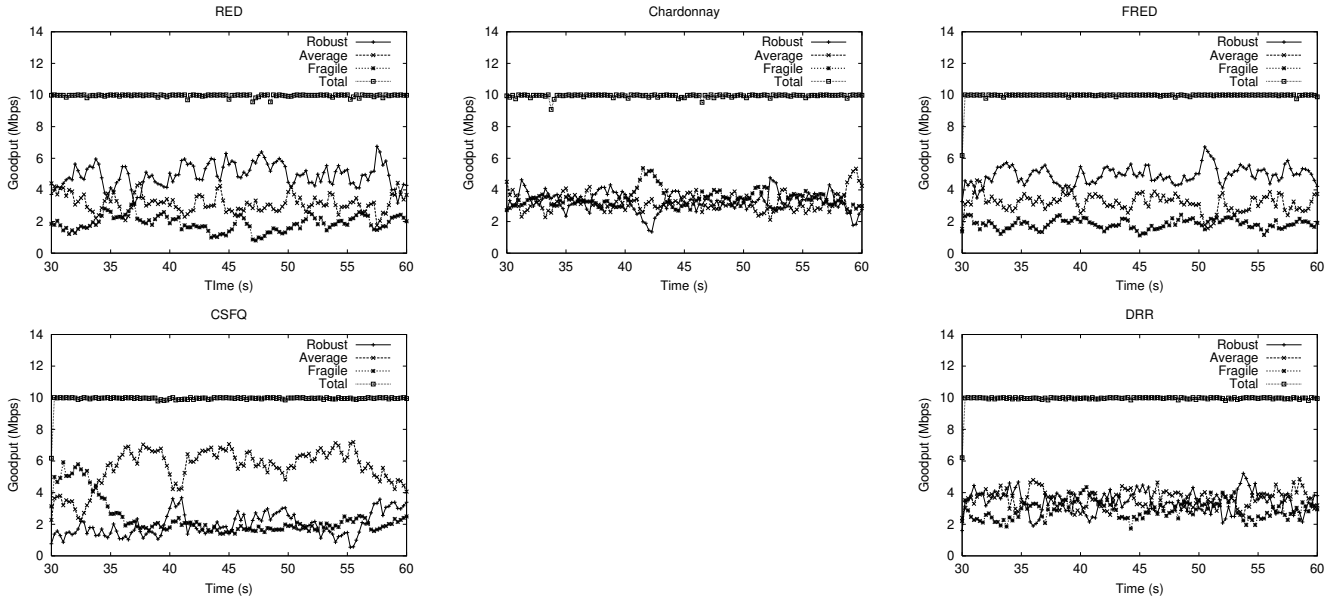

Figure 9: Experiment 2 (Balanced Clustered Latencies). 10 robust flows (50 ms round trip latency), 10 average flows (100 ms roundtrip latency), and 10 fragile flows (200 ms of roundtrip latency).

30 seconds after the beginning of each simulation. Figure 8 depicts the goodput for each flow. With 30 different flows, the fair share of each flow is $\frac{1}{3}$ Mbps, depicted by the horizontal line in each graph. Without any fair treatment, RED provides the most bandwidth (0.79 Mbps) to the most robust source and the least bandwidth (0.15 Mbps) to a fragile source. FRED does not improve the bandwidth for the most fragile flow at all (0.14 Mbps) but reduces the bandwidth of the most robust flow (0.69 Mbps). CSFQ provides high bandwidth to every other robust flow (the largest getting 0.82 Mbps), while the least bandwidth is comparable to that of RED (0.14 Mbps). Visually, DRR and Chardonnay perform much better (more sources are near the horizontal line), with DRR providing reasonable fairness between the most robust flow (0.53 Mbps) to the most fragile flow (0.20 Mbps), and Chardonnay providing the best fairness between the most robust flow (0.40 Mbps) and the most fragile flow (0.15 Mbps).

3.2 Balanced Clustered Latencies

Experiment 2 considers flows uniformly balanced in three clusters of latencies. There are 10 robust flows with 50 ms of

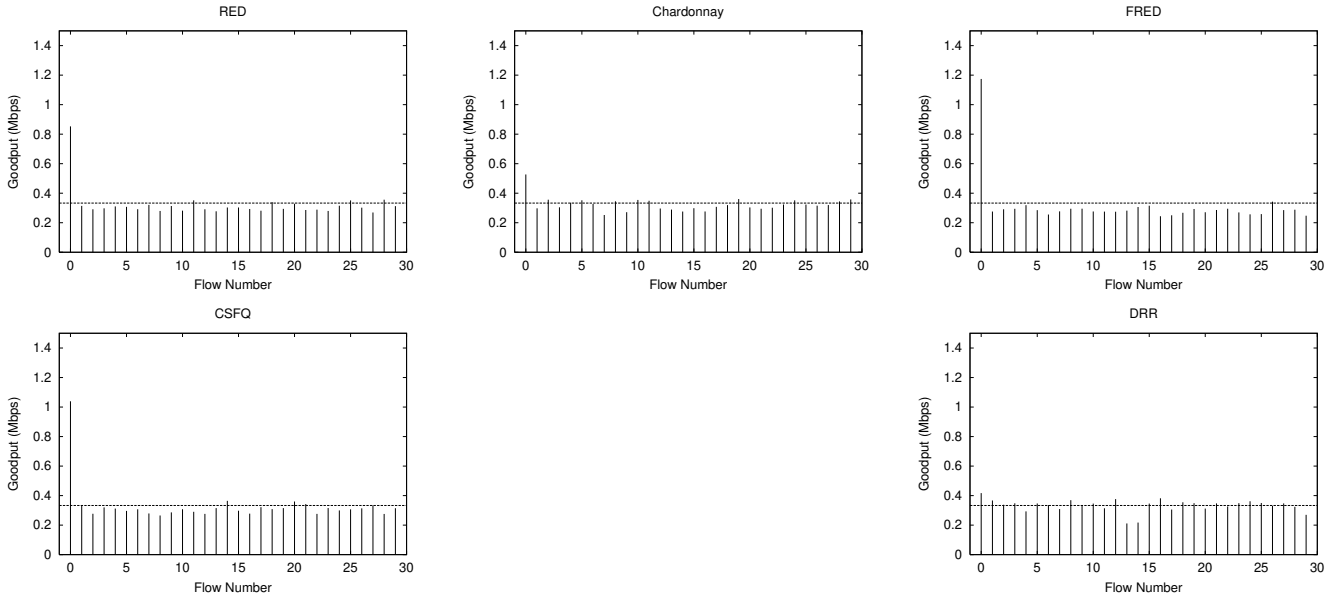

Figure 10: Experiment 3 (Unbalanced Latencies). 1 robust (flow 0, 20 ms roundtrip latency) and 29 fragile (flows 1-29, 200 ms roundtrip latency)

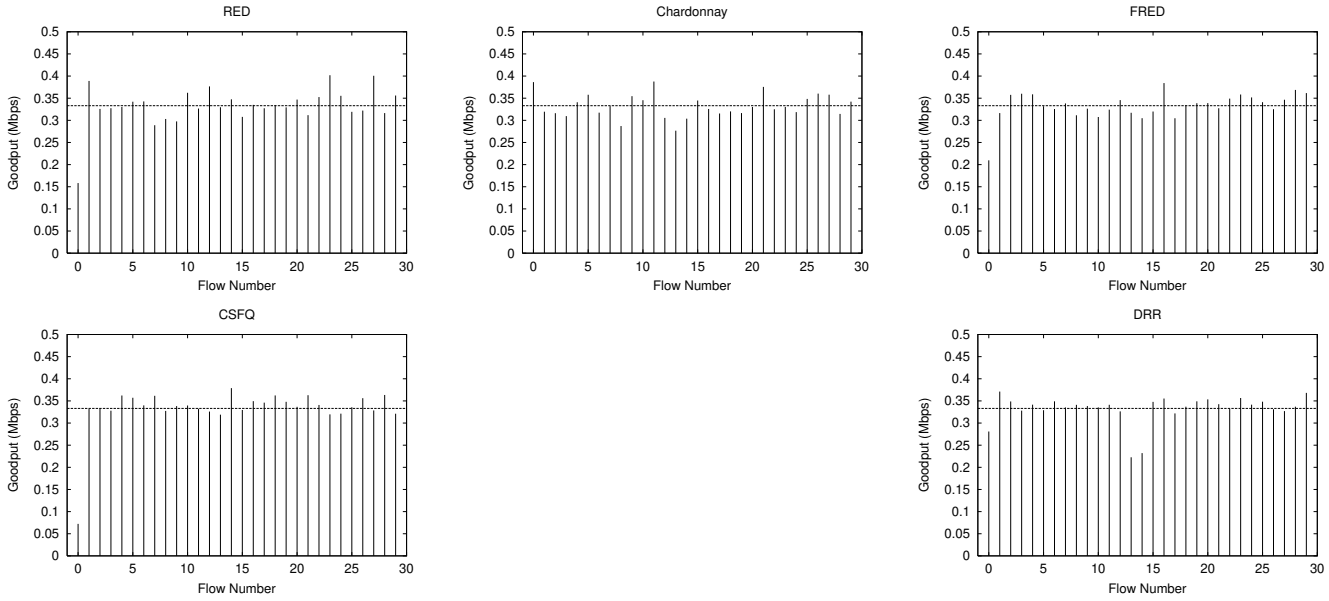

Figure 11: Experiment 4 (Unbalanced Latencies). 1 fragile (flow 0, 200 ms roundtrip latency) and 29 robust (flows 1-29, 20 ms round trip latency)

.

roundtrip latency, 10 average flows with 100 ms of roundtrip latency, and 10 fragile flows with 200 ms of roundtrip latency.

The experiment simulations were run for 150 seconds, but only the region between 30 seconds and 60 seconds is depicted to show the network in steady state. Figure 9 depicts the goodput averaged over all the flows in each cluster, measured every 250 ms. For the three clusters of flows, the fair share of bandwidth is $\frac{10}{3}$Mbps.

RED provides the fragile flows with only 1.5 Mbps while the robust flows get 6.0 Mbps. FRED improves the bandwidth of the fragile flows to 1.9 Mbps and reduces the bandwidth of the robust flows to 5.0 Mbps. CSFQ yields 2.0 Mbps to the clusters of fragile flows and 5.7 Mbps to the cluster of robust flows. DRR provides reasonably fair bandwidth allocation with the robust flows getting 3.5 Mbps and the fragile flows getting 2.8 Mbps. Chardonnay performs the best with the robust flows getting only 3.6 Mbps and the fragile flows getting 3.1 Mbps.

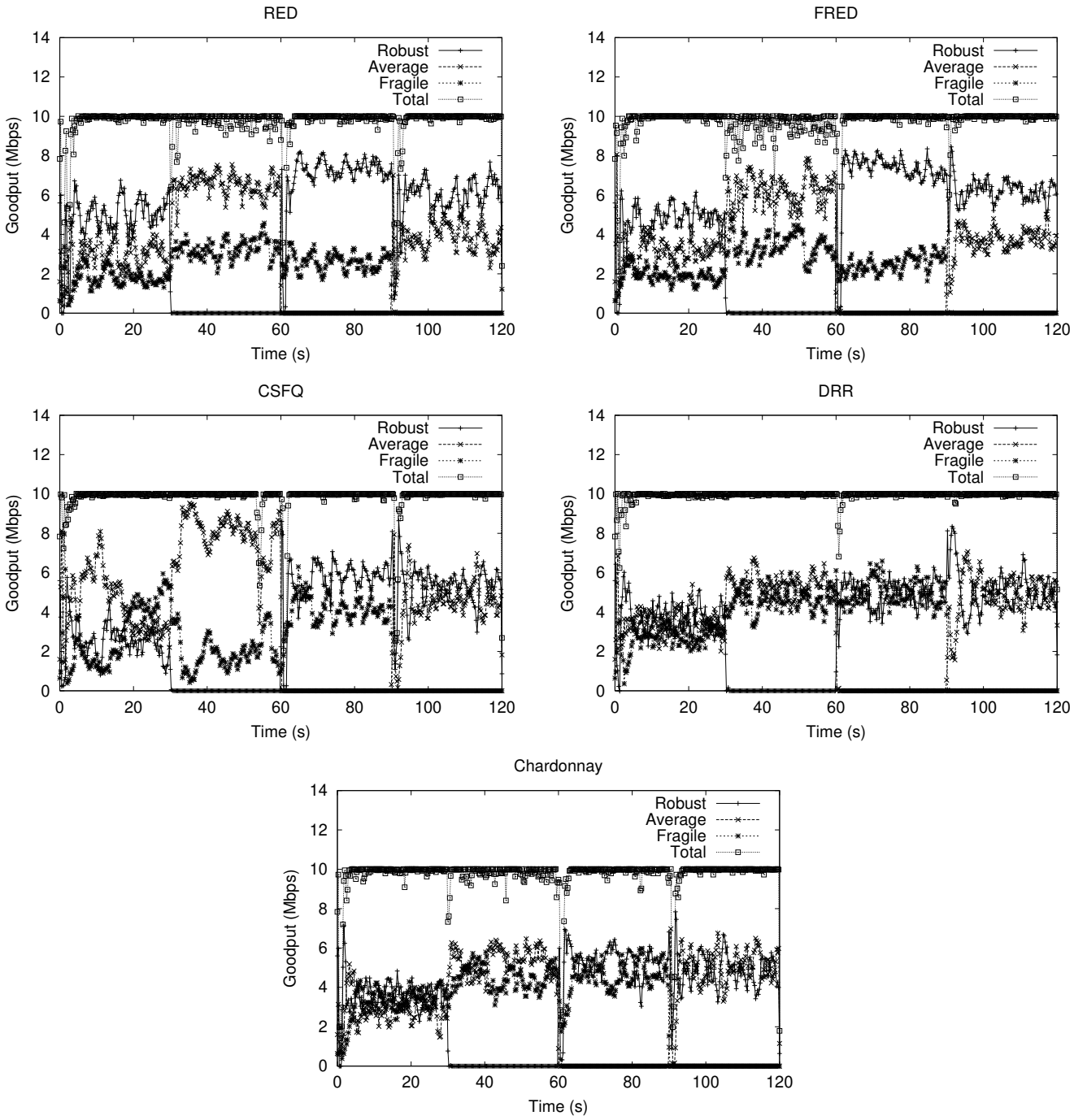

Figure 12: Experiment 5 (Dynamic Latencies). 10 robust (50 ms round trip latency), 10 average (100 ms roundtrip latency), and 10 fragile (200 ms of roundtrip latency).

3.3 Unbalanced Latencies

Experiment 3 and 4 consider cases where most (29) of the flows have the same latency, while one flow has an extremely different latency. Experiment 3 has 1 robust flow (20ms) and 29 fragile flows (200ms). Experiment 4 has 1 fragile flow (200ms) and 29 robust flows (20ms). Both experiments ran simulations for 150 seconds with the first 30 seconds omitted in order to depict a steady state. For both experiments, we compare all 30 flows individually, so the fair share of network bandwidth is $\frac{1}{3}$Mbps.

Figure 10 depicts the goodput of each flow for the simulations in experiment 3. RED fails to reduce the goodput of the robust flow (flow 0) and gives it 0.85 Mbps. FRED and CSFQ perform worse than RED, giving the robust flow 1.17 Mbps and 1.04 Mbps, respectively. DRR performs the best, restraining the robust flow to 0.42 Mbps. However, DRR

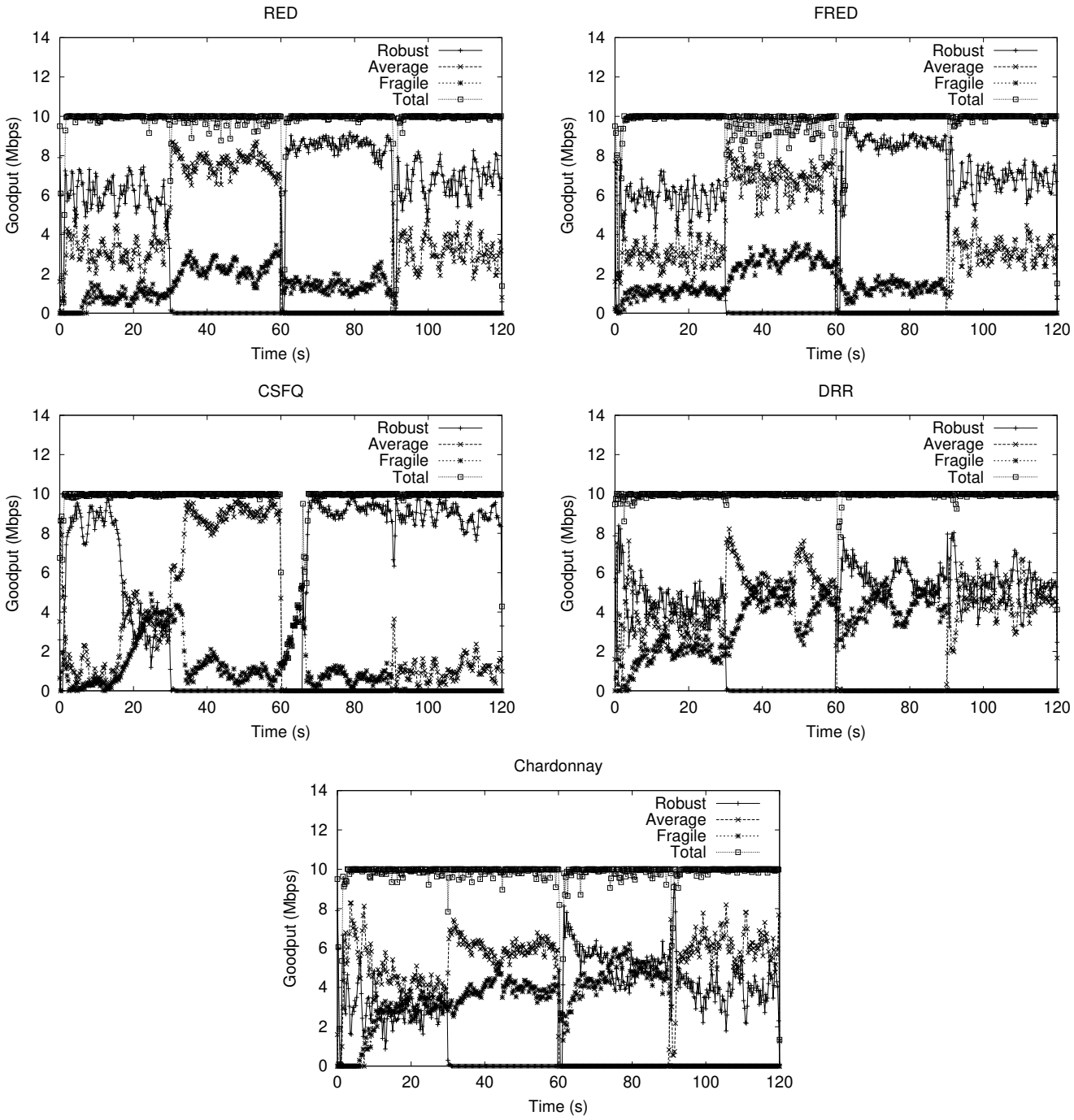

Figure 13: Experiment 6 (Dynamic Latencies). 10 robust (20 ms round trip latency), 10 average (80 ms roundtrip latency), and 10 fragile (320 ms of roundtrip latency).

gives as low as 0.21 Mbps to the fragile flows. Chardonnay comes close to DRR by only allowing the robust flow 0.53Mbps, treating the fragile flows better by giving them at least 0.36Mbps.

Figure 11 depicts the goodput of each flow for the simulations in experiment 4. RED does not assist the fragile flow, giving it only 0.16 Mbps. FRED is able to help the fragile flow get 0.21 Mbps. CSFQ performs worse than RED, giving the fragile flow only 0.07 Mbps. DRR provides fairness similar to that of FRED, giving the fragile flow 0.22 Mbps. Chardonnay provides the smallest bandwidth gap, giving the fragile flow 0.39 Mbps, slightly more bandwidth than the fair share.

3.4 Dynamic Latencies

For experiments 5 and 6, we consider abrupt changes in the latencies of clusters of flows. We set up a scenario where the 3 clusters of flows, 30 flows in each cluster, where all run for the first 30 seconds (period A). The robust flows stop for the next 30 seconds (period B). The robust flows come back and the average flows turn off for 30 seconds (period C). Lastly, the average flows turn back on and the fragile flows stop for 30 seconds (period D). When there are three clusters of flows, the fair share of bandwidth is $\frac{10}{3}$ Mbps. When there are two clusters of flows, the fair share of bandwidth is $\frac{10}{2}$ Mbps.

Experiment 5 uses the latency clusters as in Section 3.2, with 10 robust flows having 50 ms of roundtrip latency, 10 average flows having 100 ms of roundtrip latency, and 10 fragile flows having 200 ms of roundtrip latency. In experiment 6, the differences in latencies for each cluster is larger, with 10 robust flows having 20 ms of roundtrip latency, 10 average flows having 80 ms of roundtrip latency, and 10 fragile flows having 320 ms of roundtrip latency.

For experiment 5, Figure 12 depicts the goodput averaged over all the flows in each cluster, measured every 250 ms. Visually, RED, FRED and CSFQ are very unfair. Period A is exactly the same as Experiment 2 (Balanced Clustered Latencies) in Section 3.2. In period B, RED, FRED and CSFQ allows the average flows get around 6-7 Mbps while fragile flows to get only around 2-3 Mbps. Chardonnay provides fairness very close to that of DRR by giving 4.5 Mbps to the fragile flows and 5.3 Mbps to the average flows. In period C, the difference in bandwidth grows bigger for RED and FRED, with the robust flows getting around 7 Mbps and the fragile flows getting only 2.5 Mbps. CSFQ performs better than RED and FRED, giving 2 Mbps to the fragile flows and 5.6 Mbps to the robust flows. Chardonnay is again comparable to DRR by giving 4.4 Mbps to the fragile flows and 5.3 Mbps to the robust flows. In period D, RED, FRED and CSFQ provide slightly better fairness by allowing the robust flows to get around 5-6 Mbps and allowing the fragile flows to get 3-4 Mbps. Chardonnay was close to DRR, with the robust flows getting 4.7 Mbps and the fragile flows getting 5.0 Mbps.

For experiment 6, Figure 13 depicts the goodput averaged over all the flows in each cluster, measured every 250 ms. Visually, RED, FRED and CSFQ clearly demonstrate unfairness. Both DRR and Chardonnay provide slightly worse fairness than in experiment 5. During period A, DRR gave the fragile flows 1.8 Mbps and the robust flows 4.5 Mbps. Chardonnay gave the fragile flows 2.3 Mbps and the robust flows 4.5 Mbps. In period B, RED, FRED and CSFQ allow the average flows to get around 7-8 Mbps and the fragile flows to get only around 1-2 Mbps. Chardonnay provides fairness similar to that of DRR by giving 4.1 Mbps to the fragile flows and 5.7 Mbps to the average flows. In period C, RED, FRED and CSFQ are even more unfair, with the robust flows getting 7-8 Mbps while the fragile flows get only around 1 Mbps. Once again, Chardonnay is comparable to DRR by giving 4.2 Mbps to the fragile flows and 5.5 Mbps to the robust flows. In period D, RED and FRED provide a bit more fairness by allowing the robust flows to get around 6 Mbps and the fragile flows around 3 Mbps. CSFQ provides the worst fairness by only giving 1 Mbps to the fragile flows and 8.9 Mbps to the robust flows. Chardonnay is close to DRR, with around 4 Mbps for the fragile flows and 6 Mbps for the robust flows.

3.5 Overall Goodput and Drop Rate

It is important that the modifications made to RED for Chardonnay do not degrade the overall performance of RED. In addition to the fairness results for experiments 1-6, we compared the packet drop rate in steady state and the total goodput for the bottleneck link.

Experiment	RED		Chardonnay	
	Drop (%)	Goodput (Mbps)	Drop (%)	Goodput (Mbps)
1	1.85	9.65	1.80	9.59
2	2.85	9.94	2.70	9.91
3	1.45	9.65	1.46	9.67
4	3.61	9.97	3.59	9.96
5	2.90	9.73	2.56	9.76
6	2.60	9.64	2.49	9.69

Figure 14: Drop Rate and Total Goodput for RED and Chardonnay

Figure 14 shows that Chardonnay's drop rate is about the same as or slightly lower than RED's. The goodput is very comparable between RED and Chardonnay, and in some cases, Chardonnay actually provided more total goodput than RED. In the few cases where Chardonnay's total goodput is lower than RED's, it is only 60 Kbps lower.

4. SUMMARY

This paper presents Chardonnay, an active queue management approach for achieving fairness among flows with heterogeneous round trip times (RTTs). Using the distributed architecture presented in [15], packets are tagged with their minimum observed RTT, allowing the Chardonnay router to make dropping decisions based upon the RTT of each flow relative to the average RTT observed at the router. This provides the potential to protect fragile flows with a high RTT from receiving unnecessarily low bandwidth, while curtailing robust flows with a low RTT to their fair bandwidth share. Moreover, the use of the RTT tag at the network edge allows Chardonnay to avoid keeping per-flow information.

We tested Chardonnay over a range of flows and over a wide-range of RTT conditions among the flows. We find Chardonnay achieves a significant degree of fairness under all conditions. We also compared Chardonnay with CSFQ [14], FRED [9] and DRR [13], algorithms specifically designed to achieve fairness, as well as to RED [6]. In all cases, Chardonnay performs far better than RED, and in many cases, Chardonnay performs far better than either CSFQ or FRED, often performing nearly as well as does DRR. We are not aware of any other router queue management techniques that can achieve better fairness than Chardonnay without using per-flow information.

Currently, Chardonnay does not curtail unresponsive flows receiving more than their fair share of bandwidth. In fact, under Chardonnay, unresponsive flows with a high RTT will

be favored over responsive flows with a low RTT. The natural extension to Chardonnay is to combine it with a rate-based active queue management technique, such as CSFQ [14] or RED-PD [11]. High-bandwidth flows could be detected and monitored as in [11], or per packet drop probabilities could be computed based on both the bandwidth used by the flow as well as the RTT.

Perhaps the largest benefit of active queue management techniques is in marking packets, using ECN [4], as an indicator of network congestion, rather than dropping them. A new version of Chardonnay, called *Chablis*, that uses marking instead of dropping, has shown great promise in further enhancing network performance, while even further increasing fairness. We are currently in the process of evaluating the Chablis mechanism under a broader range of conditions and discovering router configuration settings appropriate for marking rather than dropping.

5. REFERENCES

[1] S. K. A. Demers and S. Shenker. Analysis and simulation of a fair queuing algorithm. pages 3 – 26, Oct 1990.

[2] M. Allman. A web server's view of the transport layer. Oct 2000.

[3] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic. pages 30 – 47, Oct 1991.

[4] S. Floyd. TCP and Explicit Congestion Notification. *Computer Communication Review*, Oct. 1994.

[5] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, Feb. 1999.

[6] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, Aug. 1993.

[7] D. T. J. Padhye, V. Firoiu and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. 1998.

[8] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley and Sons, Inc., 1991.

[9] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proceedings of ACM SIGCOMM Conference*, Sept. 1997.

[10] U. of California Berkeley. The Network Simulator - ns-2. Interent site `http://www.isi.edu/nsnam/ns/`.

[11] S. F. R. Mahajan and D. Wetherall. Controlling high-bandwidth flows at the congested routers. In *In Proceedings of the 9th International Conference on Network Protocols (ICNP)*, Nov 2001.

[12] S. Shenker. Making greed work in networks: A game theoretical analysis of switch service disciplines. pages 47 – 57, Aug 1994.

[13] M. Shreedhar and G. Varghese. Efficient Fair Queueing Using Deficit Round Robin. In *Proceedings of ACM SIGCOMM Conference*, pages 231 – 243, Sept. 1995.

[14] I. Stoica, S. Shenker, and H. Zhang. *Core*-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *Proceedings of ACM SIGCOMM Conference*, Sept. 1998.

[15] I. Stoica and H. Zhang. Providing Guaranteed Service Without Per Flow Management. In *ACM SIGCOMM Computer Communication Review , Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, Aug 1999.