Fair Bandwidth Allocation for Responsive and Unresponsive Flows Using a Capture-Recapture Model

Ming-Kit Chan and Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Email: mingkit@cs.ust.hk, hamdi@cs.ust.hk

***Abstract*—In this paper, we propose a novel technique called CARE (CApture-REcapture fair sharing) to provide fair bandwidth sharing. CARE estimates two network resource parameters: the number of flows in the buffer and the data source rate of a flow by using a *capture-recapture* model. The capture-recapture model depends on simply the random capturing/recapturing of the incoming packets, and as a result, it provides a good approximation tool with low time/space complexity. Our experiments and analysis will demonstrate that CARE provides highly accurate fair bandwidth share under different network configurations and outperforms the existing mechanisms.**

***Keywords*—Active Queue Management, Capture-Recapture Model, Fair Bandwidth Sharing**

I. Introduction

One of the challenges in the design of switches/routers is the efficient and fair use of the shared bottleneck bandwidth among different Internet flows. In particular, to provide fair bandwidth sharing, different buffer management schemes are developed to protect the well-behaved flows from the misbehaving flows. However, most of the existing buffer management schemes cannot provide accurate fair bandwidth sharing while being scalable. The key to the scalability and fairness of the buffer management schemes is the accurate estimation of certain network resources without keeping too much state information. Throughout the years, researchers have developed various buffer management schemes in an attempt to solve the fair bandwidth sharing problem, for example, Stabilized RED (SRED)[1], RED with Preferential Drop (RED-PD)[2], and Stochastic Fair Blue (SFB)[3]. However, their implementation in routers have not been realized because of scalability, quality of solution, and/or complexity problems.

On the other hand, some solutions suggest to separate flows into different queues, but such AQM solutions [7] for providing fair bandwidth sharing require routers to store per flow states, and to perform per flow operations and per flow classification. In this paper, we propose a novel AQM scheme, called CARE, that requires small bounded number of states, but can provide fair bandwidth sharing similar to those that can be provided with per flow mechanisms. In this way we can simultaneously achieve high Quality of Service, high scalability and robustness. The key technique we use is called the *capture-recapture* (CR) model, which provides an accurate estimation of the number of active flows and data source rates with the help of a random packet capturing process. A series of simulation results are provided to prove that this novel technique makes significant improvement over state-of-the-art AQM schemes.

This research is supported in part by a grant from the Hong Kong Research Grants Council under the grant RGC-HKUST6181/01E

The paper is organized as follows: the next section introduces the methodology of the capture-recapture (CR) model. Different structures of the CR model are also presented. Section III describes the mechanism of CARE in details. In Section IV, we compare the performance of CARE and the existing AQM schemes. We also present additional useful properties of CARE. Finally, a brief conclusion will be given in Section V.

II. The Capture-Repcature Model

The original objective of the capture-recapture (CR) [5] model is to estimate the number of animals in a population. Animals are first captured, marked and released. Then they are recaptured again. A number of marked animals among those recaptured determine the size of the population. In the following, we focus on the methodology based on two variants of the CR model called the M_0 CR model and the M_h CR model.

A. M_0 Capture-Recapture Model

The M_0 CR model is the most basic form of the CR model. The model assumes a constant capture probability for all the animals, where the capture probability refers to the chance of individual animals being caught. Therefore, the M_0 model assumes that the capture probabilities for all animals are the same and the effect of capture probabilty is insignificant. The model derives the estimation of the total population size as follows: Suppose that there are n_1 animals captured from the population and all of them are marked. Let n_2 be the number of recaptured animals. The M_0 CR model defined that the proportion of marked animals found among the recaptured animals is the same as the proportion of the captured animals to the population. As a result, the size of population (N) is estimated using the following equation $\frac{m_2}{n_2} = \frac{n_1}{N}$, where m_2 is the number of animals appeared to be marked among the recaptured animals.

B. M_h Capture-Recapture Model

Now we consider the case where the capture probability are different among the animals. In some circumstancess, capture probabilities may vary by animal, for reasons like differences in species, sex, or age. To achieve an accurate approximation under different capture probabilities, a new approach called M_h CR model should be used. Unlike the M_0

model, the M_h model can have as many as n+1 parameters: N and $p_1, p_2, ..., p_n$, where p_i is the capture probability for an individual animal i and N is the size of the total population. Estimating these many parameters from the capture-recapture data is not possible. In order to solve this problem, the jackknife estimator is used to estimate N without having to estimate all the capture probabilities [8]. To increase the accuracy of the estimation, multiple capture occasions are adopted. In fact, the major different between M_0 CR model and M_h CR model is the number of capture occasions performed. For the M_0 model, two capture occasions [1] are performed where the number of captures in the first capture occasion is n_1 and the number of captures in the second capture occasion is n_2. On the other hand, we could have t capture occasions for the M_h model, where the number of captures of each capture occasion are n_1, n_2, ... n_t respectively. Another difference between M_0 CR model and M_h CR model is the input parameters used. For the M_0 model, the number of captures (n_1 and n_2) are used estimate the total population. For the M_h model, however, the capture frequency data are used to ease the effort of estimating the capture probabilities $p_1, p_2, ..., p_n$. Hence, estimation of N under the M_h model is based on the capture frequency data $f_1, f_2, ..., f_t$ where f_1 is the number of animals caught only once, f_2 is the number of animals caught only twice, ... etc. In order to compute N from a set of capture frequency data, the jackknife estimator (N_{JK}) is used and it is computed as a linear combination of these capture frequencies, such that:

$$N_{JK} = a(t,K)_1 f_1 + a(t,K)_2 f_2 + ... + a(t,K)_t f_t$$

where $a(t,K)_i$ are the coefficients which are in terms of the number of capture occasions (t) and K represents the order of the estimation. In fact, the estimated process is complicated, and is intentionally omitted here in order not to put the paper out of focus. For more details, the reader is referred to [8].

III. A Buffer Management Scheme using the CR Model

In general, a fair bandwidth sharing schemes should provide the following functions: the estimation of the sending rate of individual flows, the estimation of the fair share, and the mechanism of flow rate adjustment. Based on the data rates and fair share, packets are dropped (or marked) according to the adjustment process. Hence, a scheme with an accurate estimation of the flow sending rate and an appropriate fair share guarantee a good fair bandwidth sharing mechanism. [2]

A. Estimation of the fair share by using the M_h CR model

Firstly, let us discuss the estimation of the fair share. Consider a buffer, which stores the recently arrived packets, is used for the estimation of the network parameters. Assume that all senders are aggressive enough to occupy the available bandwidth. Therefore, each flow should occupy no more than a fixed number of packets or the fair share size (S) in the buffer in order to receive equal proportion of the bandwidth. If a flow has occupied more than the fair share, packets of this flow should be dropped to leave space for the other flows. In this case, S can be calculated as $S = \frac{B}{n}$, where B is the buffer size and n is the total number of flows in the buffer.

As B is a predefined value, estimating the appropriate value of S requires the estimation of the number of flows (n) in the buffer. Based on the M_h CR model, we can estimate n by considering the total number of the flows in the buffer as the total of the animals in the population. For example, there are n flows in the buffer and x_1 represents the number of packets in the buffer having flow ID number 1, and so forth, such that the buffer should contain $x_1 + x_2 + ... + x_n$ packets. If we capture a random packet in the buffer, the chance of flow i's packet being caught is:

$$p_i = \frac{x_i}{(x_1 + x_2 + ... + x_n)}$$

where p_i denoted the capture probability of the flow i

As the data rate of different flows are different, for instance, TCP flows have fluctuate sending rate. Hence different flows may occupy the buffer by different amounts, therefore $x_i \neq x_j$ for $i \neq j$ and the capture probabilities (p_i) vary by flow. To approximate the number of flows (n) in the buffer, we choose M_h CR model as our estimation model. The estimation process using the M_h CR model is as follows:

1) Capture t [3] packets from the buffer
2) Construct a set of capture frequency data by observing the flow ID of the captured packets
3) Estimate the total number of flows in the buffer using the jackknife estimator

B. Estimation of the data rate by using the M_0 CR model

Next, we consider the estimation of source data rates. The sending rate of a certain flow can be represented by the packet counts of the flows in the virtual buffer. As a result, our goal is to estimate the number of packets belonging to a certain flow in the virtual buffer by using a capture-recapture model, particularly, the M_0 CR model. Consider the following example: Assume that all the packets with flow ID x are captured and marked when they arrive, so that n_1 is the number of captures, and it is also the number of packets in the buffer with flow ID x. B is the size of the virtual buffer. In fact, the marking procedure is not required. We may treat the mark as the flow ID in the packet header. Therefore, n_1 can be estimated by using the equation which solves the M_0 model, we modified the original equation, such that $n_1 = B \times \frac{m_2}{n_2}$, where n_2 is the number of the recaptured packets, and m_2 is the number of the marked packets among the recaptured packets. Hence, the process of estimating the number packets belongs to flow ID x is as follows:

1) Capture n_2 packets from the buffer
2) Count the number of packets with flow ID x, and let it be m_2

[1] For the M_0 CR model, the first capture occasion is referred as *capture* while the second capture occasion is referred as *recapture*.

[2] We adopte the rate adjustment mechanism of CSFQ [6]. Therefore, the dropping probabilty of flow i is $d_i = 1 - fairshare/rate_i$, where $rate_i$ is the estimated sending rate for flow i.

[3] For simplicity, we have to define the number of captures in each capture occasion $(n_1, n_2, ..., n_t)$ to be 1

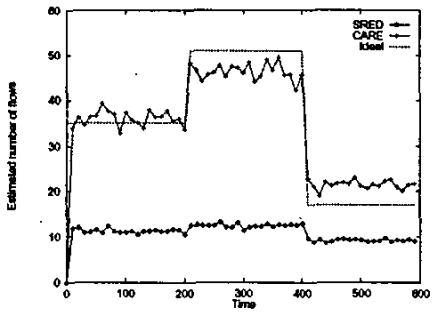

Fig. 1. Estimation of variable number of flows (30% UDP load).

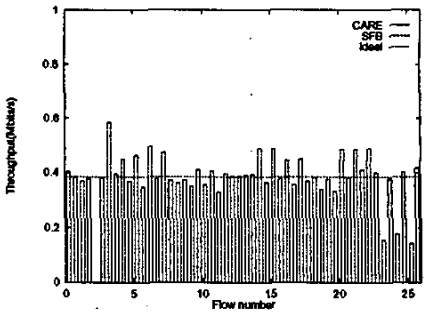

Fig. 2. Throughput fairness between CARE and SFB.

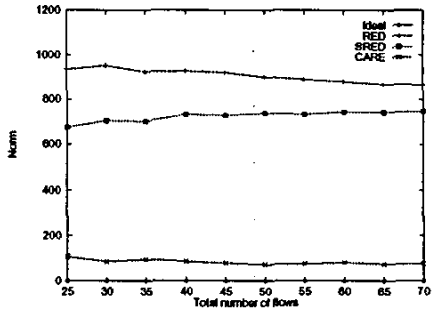

Fig. 3. Norm Analysis between CARE, SRED, and RED.

3) The estimation n_1 is calculated if the buffer size B is given

Finally, based on the previous analysis, an Active Queue Management scheme, called CARE (CApture REcapture fair sharing), is developed. Although the nature of the traditional CR model and the AQM algorithms are different, simulation results show that CARE is found to be useful in providing fair banding sharing.

IV. Performance Evaluation

A. Simulation Setup

In this section, we evaluate the performance of CARE and compare it with the existing AQM mechanisms. We use ns-2 [9] for our simulation. The network configuration is as follows: The network topology is a dumbbell. By default in our simulations, the capacity of the congested link is 10Mbps, while the link speed is 100Mbps for the others. Link latencies for all the links are 2.0msec. The packet size for all the traffic is 1000 bytes. In order to evaluate different kinds of traffic, a non-responsive constant rate flow (e.g., UDP flow) which occupies 10% of the bottleneck bandwidth is injected into the network. The UDP source has the greatest flow ID. For the parameters of CARE, the number of capture occasions is 200 (50 for the estimation of the number of flows), the number of captures per occasion is 1. We run each of the simulations for 10 minutes (600 seconds), while the results of the first 100 seconds are dropped.

B. Estimation of the number of flows

First, We evaluate CARE and SRED (SRED uses the estimated number of flows to provide fair sharing) with variable number of flows. Flows are injected and released from time to time. The simulation result shows the responsiveness of the algorithm against the change of the number of flows in the buffer. As illustrated in For Fig. 1, the CARE algorithm adjusts itself much better to traffic fluctuations than SRED.

C. Throughput Comparison

Here, we compare the throughput of each flow using CARE and Stochastic Fair Blue (SFB) in Fig. 2 . There are 70 TCP sources and 1 UDP source in the network. The result shows that the CARE algorithm provides better bandwidth sharing than SFB in terms of fairness. In fact, it is very close to the "ideal" case where complete per flow state information is needed.

In order to have a better understanding about the performance of the CARE, SRED, Stochastic Fair Blue (SFB), RED with Preferential Drop (RED-PD) and RED under different network configurations, we evaluate them with 25 to 70 TCP flows. As in the previous simulations, we added a UDP flow for each set of TCP flows. To illustrate the performance of different networking setups in a single graph, we compared the *norm* of the throughput for each algorithm. *norm* is defined as $norm = \sum(b_j - b_i)^2$, where n is the total number of flows, b_j is the throughput for flow j in kbits/sec, b_i is the ideal fair share in kbits/sec. With the norm of the ideal case being 0, the lower the value of *norm* means better performance (more fairness). Fig. 3 and Fig. 4 compare the result of the *norm* values of CARE to that of SRED, RED, SFB, RED-PD and the "ideal" case. As can be seen, the *norm* of CARE is much closer to the ideal case than the others.

D. Performance under different network setups

In the previous simulations, we set the UDP load as 10% (1Mbit/s) of the bottleneck link bandwidth (10Mbit/s). To study the effect of UDP load in our simulations, we set up the pervious simulations again with the injection of different amounts of UDP traffic. There are 35 TCP flows and 1 UDP flow for each case. Fig. 5 shows the performance of CARE and other schemes under different UDP loads. Among our samples, only RED-PD, SFB, and CARE do not suffer from increasing the load of UDP traffic. In particular, CARE performs the best among these three algorithms. Moreover, to

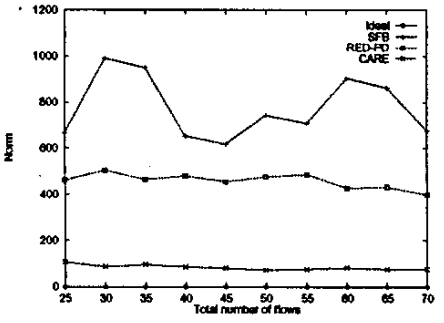

Fig. 4. Norm Analysis between CARE, SFB, and RED-PD.

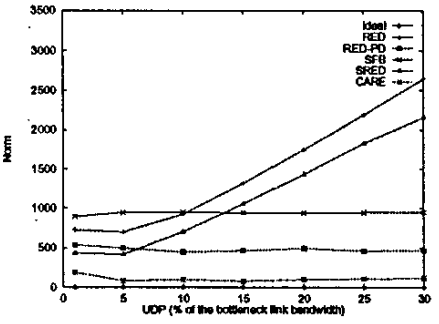

Fig. 5. Performance of CARE, RED-PD, RED, SFB, and SRED under different amount of UDP traffi c.

show the estimation using the capture-recapture model under the present of the short-lived flows, we evaluate the CARE algorithm with HTTP connectios. In Fig. 6, there are 250 HTTP short-lived connections, 100 TCP connections, and a UDP connection sending at a rate of 10Mbps. The number of capture occasions (t) is set to be 50 in this experiment. Finally, we also evaluate the algorithms using TCP with different RTTs. In this experiment, we consider 25 TCP flows and 1 UDP flows. The propagation for these TCP flows are 0.1ms, 2ms, 4ms, 10ms, and 100ms, such that flow 0 to flow 5 experience a delay of 0.1ms, and so forth. Fig. 7 shows the result. Although TCP flows experiencing high propagation delay (flow 20 to flow 24) are suffer from low throughput under the RED-PD algorithm, CARE provides improvement for these flows.

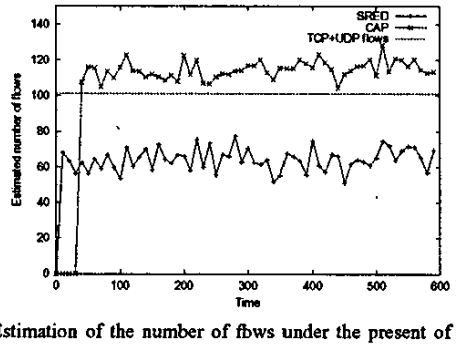

Fig. 6. Estimation of the number of fbws under the present of short-lived fbws.

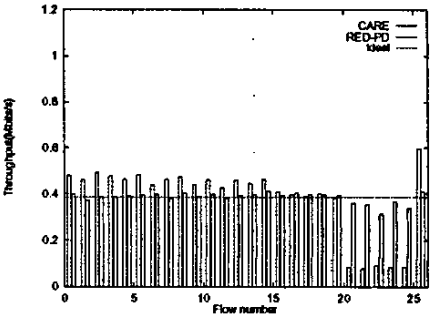

Fig. 7. Throughput of TCP with different RTTs.

V. Conclusion

In this paper we have introduced the mechanism of applying the capture-recapture model in active queue managements so as to determine crucial network resources that are needed to achieve a fair bandwidth sharing scheme. In particular, we have illustrated how to use the CR model to estimate the number of flows and the sending rate of each flow. Then these values are used for the fair bandwidth allocation among both the responsive as well the non-responsive flows. Through extensive simulations, we have demonstrated that our scheme outperforms related state-of-the-art AQM schemes. In addition, given the low complexity of this scheme, it is amenable to high-speed implementation which is crucial for possible deployment in core routers.

References

[1] T. J. Ott, T. V. Lakshman, L. H. Wong, 'SRED: Stabilized RED," *IEEE INFOCOM*, March 1999.

[2] R. Mahajan, S. Floyd, D. Wetherall, 'Controlling High-Bandwidth Flows at the Congested Routers," *9 th International Conference on Network Protocols (ICNP)*, November 2001.

[3] Wu-chang Feng; K.G. Shin; D.D. Kandlur; D. Saha "The blue active queue management algorithms," *IEEE/ACM Transactions on Networking*, 10(4), Aug 2002.

[4] S. Floyd and V. Jacobson, 'Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, 1(4):397-413, 1993.

[5] G. C. White, D. R. Anderson, K. P. Burnham, and D. L. Otis, 'Capture-recapture and removal methods for sampling closed populations," Los Alamos National Laboratory LA-8787-NERP. 235 pp., 1982.

[6] I. Stoica, S. Shenker, H. Zhang, 'Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks," *ACM SIGCOMM 1998*, September 1998.

[7] D. Lin and R. Morris, 'Dynamics of Random Early Detection," *ACM SIGCOMM 1997*, September 1997.

[8] K. P. Burnham, W. S. Overton, 'Estimation of the size of a closed population when capture probabilities vary among animals," *Biometriks*, 65(3):625-633, 1978.

[9] The Network Simulator - ns-2 version 2.1b8a, http://www.isi.edu/nsnam/ns

[10] D. Bamdal, and H. Balakridhnan, 'Binomial Congestion Control Algorithms," *Proceeding of IEEE INFOCOM 2001*, April 2001.