DELAYED CONGESTION RESPONSE PROTOCOLS

A Thesis

by

SUMITHA BHANDARKAR

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2001

Major Subject: Computer Engineering

DELAYED CONGESTION RESPONSE PROTOCOLS

A Thesis

by

SUMITHA BHANDARKAR

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

A. L. Narasimha Reddy (Chair of Committee)	Nitin H. Vaidya (Member)
Pierce E. Cantrell (Member)	Chanan Singh (Head of Department)

August 2001

Major Subject: Computer Engineering

ABSTRACT

Delayed Congestion Response Protocols. (August 2001)
Sumitha Bhandarkar, B.E., Mysore University, India
Chair of Advisory Committee: Dr. A. L. Narasimha Reddy

There has been a surge of new real-time multimedia applications over the IP network which cannot use TCP because the congestion control algorithm of TCP results in drastic variations in the sending rate, which in turn affect the user perceived quality. As a result several of these applications use UDP. The fact that UDP does not respond to congestion in any way has raised several questions about possible congestion collapse of the internet and fairness to existing TCP-based applications. Several congestion control algorithms suited for such applications have been proposed which claim bandwidth in a "TCP-friendly" manner. However all of them reduce their sending rate once and as soon as allowed by the protocol design, when a packet drop occurs. In this thesis we have proposed a novel scheme by which response to congestion is *deliberately* delayed by τ RTTs. We have provided the framework for this class of protocols in general and examined three cases in particular. For these three cases we have developed analytical models and derived the conditions under which they are TCP-friendly. With these conditions we have run simulations on the ns-2 platform to show that they are indeed TCP-friendly. By showing that delayed congestion response is possible, we have laid the ground work for the development of a whole new class of protocols which are not just TCP-friendly and capable of providing early warning to the application regarding an impending reduction in the sending rate but also can be designed to have a smooth congestion response.

To my parents

ACKNOWLEDGMENTS

I am deeply indebted to my advisor, Dr. Reddy, for having given me a chance to work with him. He has been a source of inspiration to me all along. Throughout my research he has provided me with immense guidance and encouragement. He has always been very open to questions and answered them patiently. His calm, assured manner has been very motivating, especially when things weren't going as expected. He organized informal group meetings and created a conducive environment for us to discuss our work with the other students working in the same area. Thank You, Dr. Reddy, for everything.

I would like to thank Dr. Vaidya and Dr. Cantrell for their interest in this research and for their invaluable suggestions.

I would like to thank my parents and sisters for believing in me. Without the confidence they have shown in me, I would not be able to pursue my MS or this thesis. They were, are and will always be the driving factor in all my endeavors. Last but not least, I would like to thank Hemanth for being so patient and supportive, without which none of this would be possible.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Even in the face of a tremendous growth in the number of users and applications for the internet, the internet itself has been very stable. This stability has been primarily due to congestion avoidance mechanisms of TCP [1], which is still the most dominant protocol on the internet. However, for some emerging applications such as streaming and real-time audio and video, TCP is not very well suited because the congestion avoidance algorithm in TCP reduces the sending rate by half, when congestion is detected in the network. This appears to be unnecessarily severe to most multimedia applications as it can noticeably reduce the user-perceived quality [2]. Also, the reliability mechanism in TCP uses retransmission of lost packets, which results in an increased end-to-end delay. As a result, most of the real-time multimedia services such as IP-telephony and group communication choose not to use TCP. UDP, which does not provide any reliability or congestion control mechanisms seems like a good alternative in such cases.

As can be expected, the increase in the use of UDP will threaten the stability of the best-effort internet and might result in a "congestion collapse" where an increase in the network load results in a decrease in the useful work done by the network [3]. In addition to a concern about congestion collapse, there is a concern about "fairness" for best-effort traffic. Because TCP "backs off" during congestion, a large number of TCP connections can share a single, congested link in such a way that bandwidth is shared reasonably equitably among similarly situated flows. The equitable sharing of bandwidth among flows depends on the fact that all flows are running compatible

---

The journal model is *IEEE Transactions on Automatic Control.*

congestion control algorithms. The growth of best-effort traffic that does not use TCP underscores this concern about fairness between competing best-effort traffic in times of congestion.

In the light of these problems there has been a surge of interest in a new class of protocols termed as "TCP-friendly" protocols. TCP-friendliness indicates that the new protocol should simply choose to send at a rate no higher than a TCP connection under similar conditions of round trip delays and losses. Several "TCP-friendly" protocols have been proposed for streaming multimedia applications which provide stable sending rates that acquire the same bandwidth as TCP, when averaged over time intervals of several seconds or even the entire life time of the flow [4]. In this thesis we have tried to understand TCP-friendliness and the factors that affect it. We examine TCP-friendliness from not just the perspective of a protocol but also the various conditions prevailing in the network. Also, most of the currently proposed protocols respond to congestion by instantaneously reducing the sending rate, based on the conception that this is required to bring the buffer sizes back to their normal steady state values. In this thesis, by implementing Delayed Congestion Response protocols, we show that the response to congestion can be delayed over τ RTTs or smoothed over this time range.

A. Related Work

An analytical model for TCP throughput was developed by J. Padhye et al. [5] which provides the equation for TCP throughput in terms of loss rate and RTT. This model captures the behavior of fast retransmit and the timeout mechanism of the TCP congestion control algorithm and has been evaluated using the network traces obtained from the internet. According to this model the throughput T of a TCP connection

with packet size of S, RTT of R and drop probability p, is given by

$$T = \frac{S}{R\sqrt{\frac{2p}{3}} + T_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

where T_{RTO} is the retransmission timeout interval.

This can be simplified to the form

$$T = \frac{\sqrt{\frac{3}{2}}}{R\sqrt{p}}$$

Based on this equation, the criteria for TCP-friendliness is developed. Any protocol claiming to be TCP-friendly should have an average throughput which is less than or equal to the throughput indicated in the equation above. In a very generalized manner, a protocol is said to be TCP-friendly if its throughput $\alpha\frac{1}{\sqrt{p}}$

Several TCP-friendly protocols have been proposed, some that directly use the above equation and others which are based on the $\frac{1}{\sqrt{p}}$ relation. In the next few pages we provide a quick review of three such schemes.

1. TFRC : Equation based congestion control

In [6] Floyd et al. proposed a technique for implementing TCP-friendly congestion control for unicast applications which does not utilize TCP at the transport layer. Instead, the sender adapts its sending rate, guided by the TCP control equation, in response to the feedback from the receiver. The primary goal of this approach is to avoid aggressively seeking and using the available bandwidth, and maintaining a relatively steady sending rate, while still being responsive to congestion.

Applying the TCP response function using the equation above requires that the parameters R, S, T_{RTO} and p are determined. The round trip time R can be measured at either the receiver or the sender and based on it the T_{RTO} can be estimated. Also,

the packet size S is known. The receiver computes the loss event p using the Average Loss Interval method, where the loss rate is computed by averaging the loss rates over several loss intervals. Also, weighted averaging is used with larger weights for the most recent intervals, to deploy *history discounting* which allows a more timely response to a sustained decrease in congestion. In cases of persistent congestion, TFRC requires about four to eight round-trip times to halve its sending rate.

The TCP equation shown above is for the steady state and does not address the slow start behavior of TCP, where sending rate is doubled for each RTT. Traditional TCP algorithm is ACK-clocked and thus ensures that the sending rate cannot exceed twice the bottleneck link bandwidth during slow start. In order to provide similar behavior to TFRC, the receiver feeds back the rate that packets arrived at the receiver during the last measured RTT. If loss occurs then slow start is terminated, otherwise the sender sets its rate to: $T_{actual,i+1} = min(2T_{actual,i}, 2T_{received,i})$

Through simulations on ns and implementation on the Dummynet network emulator, this paper has shown that TFRC is fair when competing with TCP traffic across a wide range of network conditions. Also, for loss rates less than about 9%, the inter-flow fairness of individual TFRC is better than TCP flows. TFRC also continues to be fair in the presense of long-duration background traffic as well as ON-OFF flows.

2. LDA+ : Loss delay based adaptation algorithm

In the above mentioned protocol, the sender adapts its sending rate according to the TCP equation based on the feedback from the receiver. An alternate method was designed by Sisalem et al. in [7] which used real time transmission protocol (RTP) for exchanging feedback information about the round trip time and losses at the receiver. RTP is an application level protocol for multimedia services where data is sent by

adding an additional header to the data stream and control reports are sent periodically by using the control part of the protocol, termed RTCP. LDA+ is an additive increase, multiplicative decrease algorithm in which the addition and reduction values are determined dynamically based on the network condition in accordance with the TCP equation using RTCP messages, which include information about the losses and delays noticed in the network.

Via simulations, it has been shown that LDA+ is TCP-friendly and also that adding LDA+ flows to the network does not affect the performance of the WWW traffic any different than long lived TCP connections. Results also suggest that LDA+ reduces the network congestion while maintaining high network utilization levels. Experiments were also conducted on the internet with a host sending data packets over a TCP connection to some destination, while simultaneously the host sent UDP packets to the same destination with the transmission rate determined using LDA+. Results showed that in the presense of asymmetry in the network where the reverse channel had a higher droprate, LDA+ scheme could utilize the network much better than TCP. This was because the LDA+ adaptation algorithm depends on only the loss rate from the sender to the receiver, whereas TCP being ACK-clocked depends on reverse channel drop rates too.

Being an application level protocol which provides closed feedback loop between the sender and receiver, LDA+ is an attractive option since it does not require the implementation of transport layer protocol. However, since it depends upon the RTP protocol, and the interval between sending two RTCP messages is typically around five seconds, the RTP sender cannot change the sending rate fast enough in the case of rapid changes in the network conditions. Also, since the receiver reports use eight bits for indicating loss information, the lowest loss information that can be conveyed is 0.002. For loss probabilities, less than these LDA+ will be much more aggressive

than TCP, since the loss probability will be rounded off to zero.

3. TCP-friendly protocols based on binomial algorithms

In a slightly different perspective from the above mentioned approaches, a novel way to implement TCP-friendly protocols was proposed by Bansal et al. in [8]. In this paper, the increase and decrease rules for TCP congestion response were generalized in the form of binomial equations. Detailed analysis of these equations were made and compared with the TCP analytical model, thus presenting the conditions under which several TCP-friendly algorithms could be generated. This paper also provides the simulation results of the implementation of two such binomial algorithms and brings out the role of buffer management schemes in maintaining the TCP-friendliness of the new schemes.

The binomial equations for the TCP congestion control were written as:

$$\begin{aligned} I: \quad w_{t+R} &\leftarrow w_t + \frac{\alpha}{w_t^k}; \quad \alpha > 0 \\ D: \quad w_{t+\delta t} &\leftarrow w_t - \beta w_t^l; \quad 0 < \beta < 1 \end{aligned}$$

where I refers to the increase in window as a result of receipt of one window of acknowledgements in a RTT and D refers to the decrease in window on detection of a loss by the sender, w_t the window size at time t, R the round-trip time of the flow and α and β are constants. Also, a linear increase in the window in a RTT is assumed. With $k = 0$ and $l = 1$ these equations represent AIMD used in TCP. With k = -1, l = 1 we get MIMD, for k = -1, $l = 0$ we get MIAD and for $k = 0$, $l = 0$ we get AIAD, thereby covering the class of all linear controls. This family of algorithms where the control expressions are represented as the addition of two algebraic terms with different exponents has been named *binomial* congestion control algorithms. These protocols are simple to implement and analyze and have the additional benefit that

with $l < 1$ the decrease is in general, less than a multiplicative decrease which is a desirable property for streaming and real-time audio and video.

On the analysis of these equations it was found that the throughput T α $1/p^{\frac{1}{k+l+1}}$. In order that this family of protocols is TCP-friendly, it should satisfy the relationship T α $1/\sqrt{p}$. Thus we have the $k + l$ rule, namely, $k + l = 1$ and $l \leq 1$ for suitable values of α and β. This implies that there is a wide range of TCP-friendly binomial controls parametrized by k and l, and applications can choose from this family depending on their needs and the level of rate degradation they can sustain. Also the binomial control protocols converge to fairness as long as $k \geq 0$, $l \geq 0$ and $k + l > 0$.

Simulations were conducted and results were presented for two interesting TCP-friendly algorithms named as IIAD (inverse-increase/additive-decrease) and SQRT with ($k = 1$, $l = 0$) and ($k = 1/2$, $l = 1/2$) respectively. The results indicate that both IIAD and SQRT interact well with TCP AIMD across a wide range of network conditions over a RED bottleneck gateway.

One important observation made in this paper is that even if the a protocol adheres to the T α $1/\sqrt{p}$ rule, it may not necessarily mean that it is TCP-compatible. This indicates that over a wide range of parameters, TCP-friendliness does not necessarily imply TCP-compatibility. The unfairness stems from the buffer management algorithms implemented at the congested gateway and the size of the buffers. The less aggressive TCP-friendly protocols reduce their window, on congestion notification, less drastically than the AIMD protocols. As a result the queue length at the congested router may not be sufficiently reduced and other competing flows may be forced to have higher droprates. Thus the droprates observed by two flows competing at a droptail bottleneck are not equal, invalidating the results of the steady state analysis. In contrast, the RED gateways are designed to explicitly equalize packet

loss rates across the flows, and thus alleviate this problem.

B. Motivation

With the increase in the number of real-time streaming audio and video applications on the internet, the need for TCP-friendly protocols cannot be stressed enough. There has been a subsequent amount of research in the area of TCP-friendly protocols some of which were mentioned above. A website dedicated to TCP-friendliness [9] maintains an updated list of newer protocols. Additional references to work in related areas is provided in [10] through [19]. The primary aim of most of the TCP-Friendly protocols is to provide smooth congestion response. All these protocols respond to congestion by decreasing the sending rate once, as instantaneously as allowed by the protocol design. In this thesis, we aim to show that it is possible to have a *delayed congestion response* protocols which delay the congestion response by an interval of τ RTTs upon the notification of congestion in the network. We investigate the possible courses of action for the congestion window during this time period τ RTTs, and analyze them to develop analytical steady state models. With such a protocol which is TCP-friendly, not only do we have a choice of providing the application with an early warning of an impending reduction in the sending rate but can also provide smoother congestion response compared to TCP by choosing proper parameters. In the course of studying Delayed Response Protocols, we aim to obtain a better understanding of the concept of "TCP-friendliness".

C. Organization

The rest of the thesis is organized as follows. In chapter II, we take a look at the delayed congestion response protocols in general. In chapter III, we examine an algorithm for delayed congestion response where the window continues to increase

during the delay interval. In chapter IV, we look at an algorithm where the congestion window is decreased in smooth steps during the delay interval. In chapter V we examine an algorithm where the window is held constant during the delay interval. Finally in chapter VI, we present the conclusions and future work.

CHAPTER II

DELAYED CONGESTION RESPONSE

A. Why Study the Delayed Congestion Response Protocols ?

For the multimedia applications to be able to use an underlying protocol that responds to congestion, one of the primary requirements is that the protocol should have a smooth congestion response. Alternatively, if the transport protocol is capable of providing an advance warning of a reduction in the sending rate, then smart applications can be built to take advantage of this warning and use buffering techniques to damp out variations in the sending rate. The primary mechanism by which a protocol perceives the congestion in the network is through packet drops. The current protocols which respond to congestion normally reduce their sending rate as soon as they realize that a packet is dropped, thus leaving very little room for a warning to be sent to the application. With delayed congestion response, protocols could send a warning to the application when a packet drop is perceived, and then within a certain time interval respond to congestion.

There has not been much work in investigating the effects of delayed congestion response in contemporary literature. With the traditional routers which normally implement a droptail algorithm, congestion notification is provided to the end hosts only when the queues at the bottleneck link are filled up. In such a situation, it becomes necessary to respond to congestion immediately or atleast start the reduction in sending rate immediately, to ensure that the droprates seen by competing flows does not go up drastically. Many new active buffer management schemes (ex. RED, LQD, SFQ etc) have been proposed and RED is being provided as an option with many of the new routers. As the internet moves towards widespread deployment of routers

with active queue management schemes, it becomes possible for delayed response protocols to co-exist with protocols responding to congestion immediately. However, in order to ensure that the fairness is maintained between such flows and those that respond to congestion instantaneously, it is necessary that the delayed congestion response protocols have to be TCP-friendly. Also, since they will primarily be used with real-time streaming audio and video applications, it is necessary to keep their congestion response as smooth as possible.

From this perspective, the study of feasibility of delayed congestion response protocols, their effects on the existing protocols and the way they react to network conditions is not just a very necessary exercise, but also paves way for a whole new line of research.

B. Characterization of Delayed Congestion Response Protocols

In order to understand the delayed congestion response protocols, we represent the congestion control in terms of a quartet $(f_1(t), f_2(t), \tau, \gamma)$.When there is no congestion in the network, the protocol seeks out available bandwidth using the function $f_1(t)$. On congestion notification, the protocol delays the congestion response by τ RTTs. During this period, the congestion window follows the function $f_2(t)$. At the end of the interval τ the congestion window will be reduced by a factor γ from its current window value. Fig. 1 shows these parameters.

In Fig. 1 only one function is shown for $f_1(t)$ and three different for $f_2(t)$. The reason for this is, we conducted our experiments with three different response behaviors for the time interval τ RTTs, one where the window continues to increase, one where the window is held constant and one where the window gracefully decreases during the interval τ. The function $f_1(t)$ was kept similar to that of TCP.

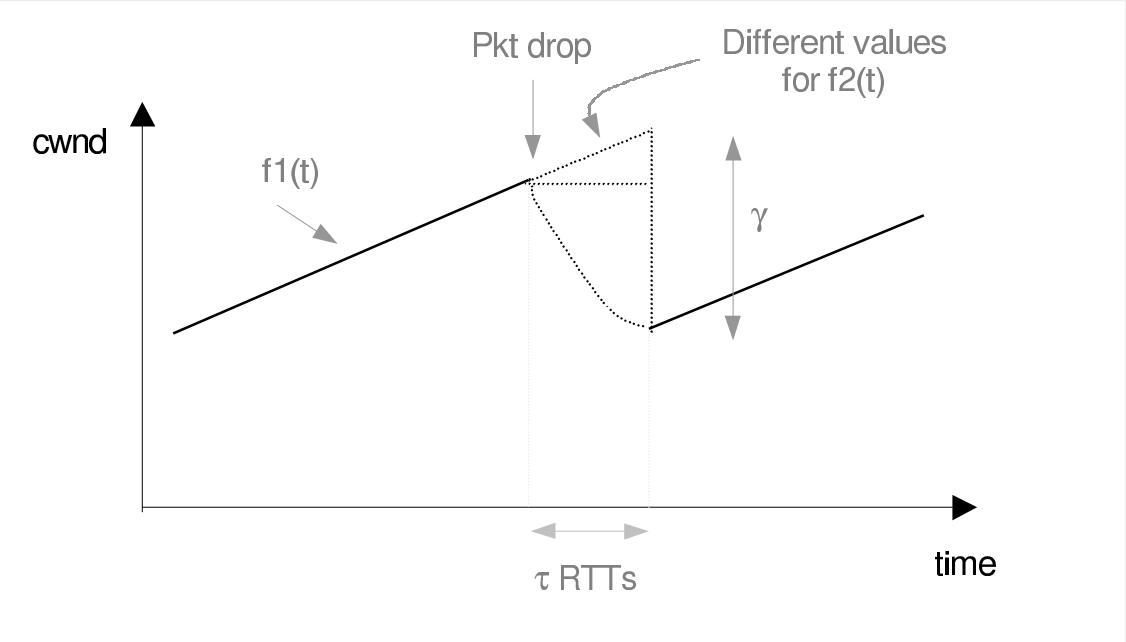


Fig. 1. Congestion Window for Delayed Congestion Response Protocols

Implementation details, analysis and the simulation results are provided in the following chapters.

CHAPTER III

DCR-I: USING AN INCREASING FUNCTION FOR $f_2(t)$

In this algorithm, the congestion window continues to increase for a time interval of τ RTTs after the congestion notification. At the end of this interval the window is reduced to γ times the current window size. The function $f_1(t)$ is similar to TCP function and the function $f_2(t)$ should be an increasing function. For the sake of simplicity, we have set $f_2(t) = f_1(t)$. Thus we have,

$$\begin{aligned} f_1(t) \quad &: w_{t+R} \leftarrow w_t + \alpha; \quad \alpha > 0 \\ f_2(t) \quad &: w_{t+R} \leftarrow w_t + \alpha; \quad \alpha > 0, t_{drop} < t < t_{drop} + \tau \\ &w_{tdrop+\tau} \leftarrow \gamma * w_{tdrop+\tau-\epsilon}; \quad \gamma < 1 \end{aligned} \tag{3.1}$$

A. Simple Analysis

We first analyzed this protocol to check if is TCP-friendly. Here we present some details of the analysis. The analysis follows closely the analysis of binomial equation protocols presented in [8]. Using the equation (3.1), we get using a continuous fluid approximation and linear interpolation of the window between w_t and w_{t+R}

$$\begin{aligned} \frac{dw}{dt} &= \frac{\alpha}{R} \\ \Rightarrow \quad w &= \frac{\alpha t}{R} + C \end{aligned} \tag{3.2}$$

As can be seen from Fig. 2, at steady state, the parameters T_D and N_D are independent from time shifting the curve along the horizontal (time) axis. This implies that one can arrange it such that a downward extrapolation of the curve passes through the origin. That is, without loss of generality and with no change to

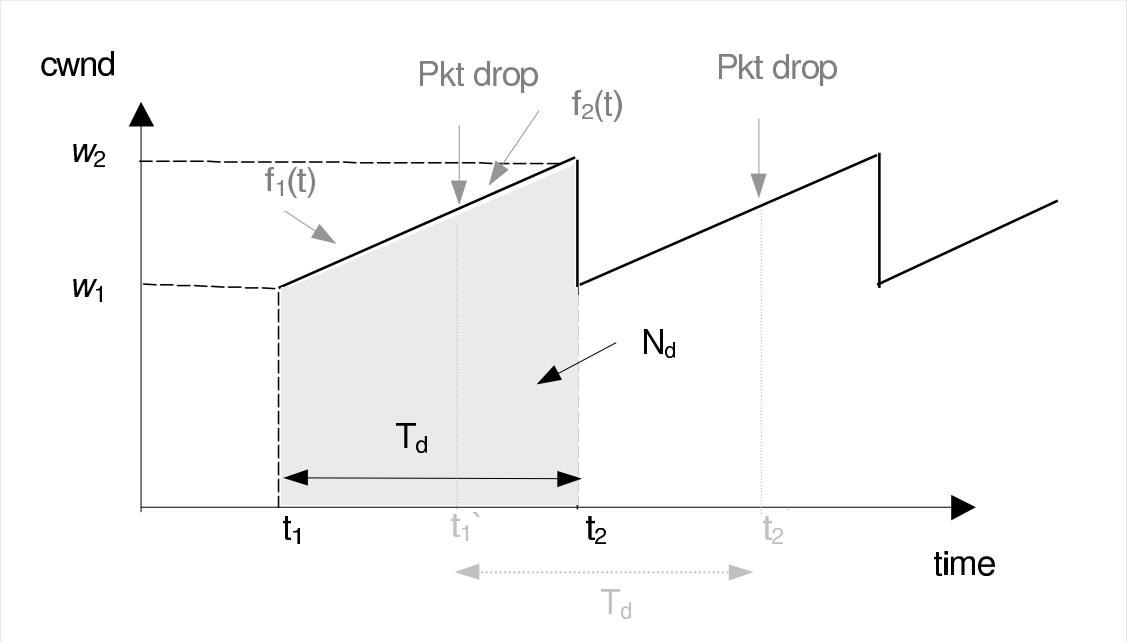


Fig. 2. Congestion Window for DCR-I Protocol

T_D and N_D, one can set $C = 0$. Thus we have,

$$\begin{aligned} w &= \frac{\alpha t}{R} \\ or \quad t &= \frac{wR}{\alpha} \end{aligned} \tag{3.3}$$

Clearly, the throughput (in packets per second), λ can be given by the number of packets that can be sent between two successive drops (N_D) divided by the time interval between two successive drops (T_D).

From Fig. 2 we have

$$\begin{aligned} T_D &= t_2 - t_1 \\ &= \frac{R}{\alpha}(w_2 - w_1) \end{aligned} \tag{3.4}$$

From the quartet defining the delayed congestion response equations we have the

parameter γ which defines the relationship between w_1 and w_2. Thus, we have $w_1 = \gamma w_2$. Substituting this in the above equation, we get

$$T_D = \frac{R}{\alpha} \cdot w_2 \cdot (1 - \gamma) \tag{3.5}$$

N_D is the shaded are under the curve in the figure . Thus we have,

$$\begin{aligned} N_D &= \int_{t_1}^{t_2} w(t) \frac{dt}{R} \\ &= \frac{1}{R} \int_{t_1}^{t_2} \left(\frac{\alpha t}{R} \right) dt \\ &= \frac{\alpha}{2R^2} \left(t_2^2 - t_1^2 \right) \\ &= \frac{1}{2\alpha} \cdot w_2^2 \cdot \left(1 - \gamma^2 \right) \end{aligned} \tag{3.6}$$

However, since N_D is the number of packets between two consecutive drops we have the steady state drop probability $p = 1/N_D$.

$$\begin{aligned} N_D &= \frac{1}{p} \\ \frac{1}{2\alpha} \cdot w_2^2 \cdot \left(1 - \gamma^2 \right) &= \frac{1}{p} \\ Thus, \quad w_2 &= \sqrt{\frac{2\alpha}{p\left(1 - \gamma^2\right)}} \end{aligned} \tag{3.7}$$

Substituting these values for the throughput equation we get,

$$\lambda = \frac{\sqrt{\frac{\alpha(1+\gamma)}{2(1-\gamma)}}}{R\sqrt{p}} \tag{3.8}$$

Thus $\lambda \quad \alpha \quad 1/\sqrt{p}$, showing that delayed congestion response protocols with the quartet ($f_1(t)$, $f_2(t)$, τ, γ) that we have taken should be TCP-friendly. In order to ensure that the throughput of a flow using this protocol is the same as that of TCP, we need

to have the following condition satisfied

$$\begin{aligned} \frac{3}{2} &= \frac{\alpha\,(1+\gamma)}{2\,(1-\gamma)} \\ \alpha &= \frac{3\,(1-\gamma)}{(1+\gamma)} \end{aligned} \tag{3.9}$$

There are an infinite number of values for α and γ that satisfy the above condition. We take $\alpha = 1$ and $\gamma = 1/2$ since this will make the control dynamics of the protocol similar to TCP-Reno, except that the congestion response is delayed by a factor of τ RTTs.

B. Implementation

In order to test the effects of such a protocol, we implemented it on the ns-2 platform. TCP-Reno was modified such that, when three dupacks were received to indicate a packet drop, instead of reducing the congestion window, the response time was noted as '$t + \tau$' RTTs. At the end of this time interval the window was reduced to half the window size that it had achieved. Thus we have $\alpha = 1$ and $\gamma = 1/2$.

C. Simulation Results

All simulations were conducted with the dumb-bell topology as shown in the Fig. 3 with one bottleneck link connecting 'n' number of sources with 'n' number of sinks. The bandwidth of the bottleneck link is BMbps and its delay is $d$ms. The buffer size at the router R1 was set to $3 * B * d$ unless specified otherwise. The routers were configured to have the RED buffer management scheme unless specified otherwise. The RED parameters of minimum threshold, maximum threshold and p_{max} were set to 25% of the buffersize, 75% of the buffersize and 0.1 respectively. A *flow* consisted of packets along the link from *src i* to *sink i*. In all the simulations half of the flows were

TCP-Reno and the other half were DCR unless specified otherwise. All simulations were conducted for 600 seconds. All the readings were gathered only after the first 100 seconds, in order to ensure that steady state was reached. Also, the bottleneck link utilization is always more than 90%, most of the time more than 99.9%.

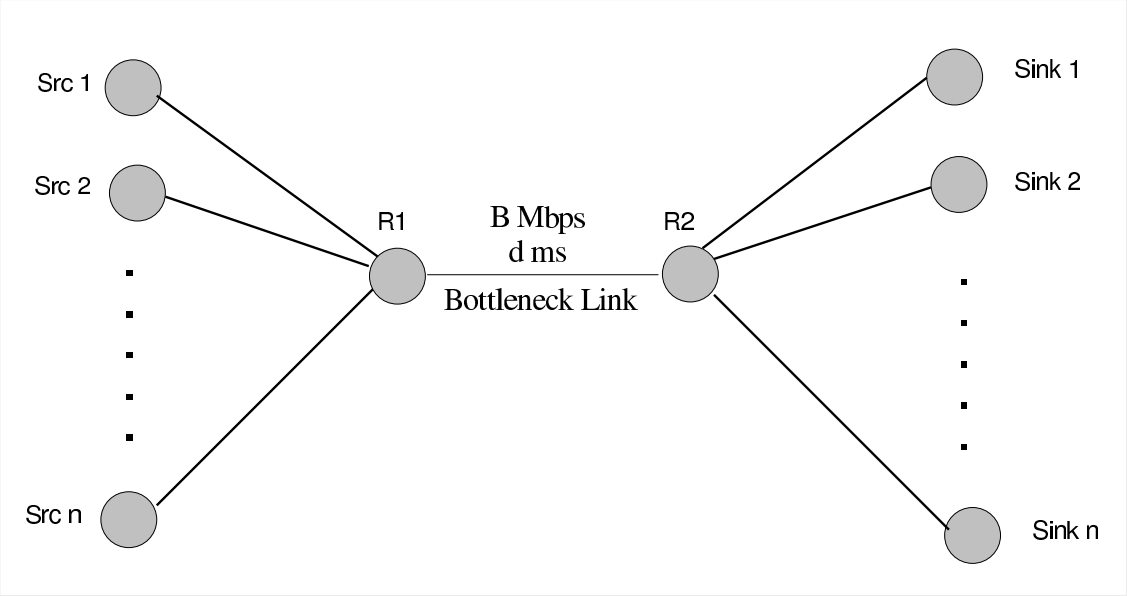


Fig. 3. Network Topology for the Simulations

The results of almost all the experiments are presented in the form of a graph. A brief description of these graphs is warranted. Each graph contains a series of curves. On the X-axis we plot the response delay and on the Y-axis we show the Fairness Index which is measured as the ratio of the average steady-state goodput of DCR flows to that of the TCP-Reno flows. Each curve represents one set of simulations for one particular value of the condition we are checking. For instance if we are verifying the Fairness Index at different droprates, then each of the curves in the graph represents the results of simulations for one value of the droprate. The accompanying legend will keep track of what the value of the condition was for which each of these curves were

obtained. In this example, the legend represents the different droprates for which the simulations were conducted and the results are depicted.

1. Fairness Index at different droprates

In this simulation, we investigate the TCP-friendliness of DCR-I at different droprates. The bandwidth of the bottleneck link was set to 10Mbps and RTT was 36 ms. Number of flows were varied to obtain different droprates. Fig. 4 shows the Fairness Index of DCR-I with respect to TCP-Reno at different values of the parameter τ for different droprates.

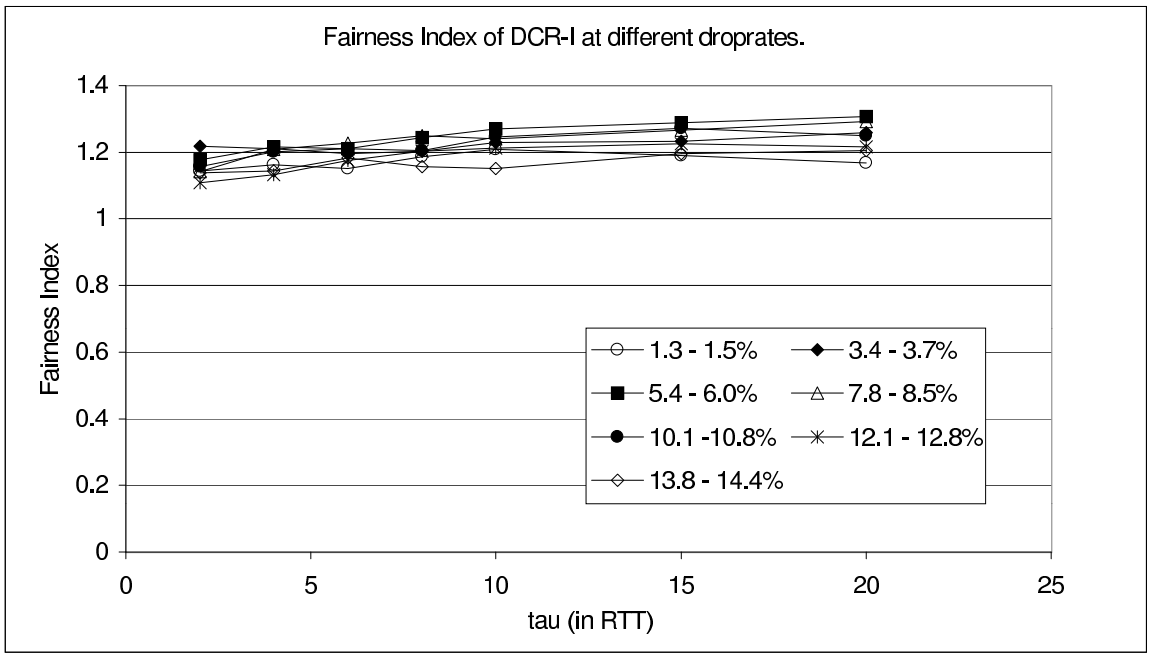


Fig. 4. DCR-I: Fairness Index at Different Droprates

It may be noted from this graph that for any particular value of τ, even while the droprates are varied, the variation in the Fairness Index remains within a 20% band except at low values of droprates. Also, as the value of τ is increased, the fairness

decreases since the Fairness Index keeps increasing. However, even for large values of τ such as 20 RTTs, the Fairness Index is 1.5 indicating that on an average, DCR-I flows get about one and half times the bandwidth share of TCP-Reno. This is fairly reasonable because even with just TCP-Reno flows, the Fairness Index is not exactly 1 for the different competing flows.

2. Fairness Index at different buffer sizes

This simulation was conducted to investigate the effects of network conditions on the TCP-friendliness of DCR-I. The bandwidth of the bottleneck link was set to 4Mbps and RTT was 60 ms. There were 8 flows in the topology out of which 4 were TCP-Reno and the other 4 were DCR-I. The buffersize at router R1 was varied from 0.5 times to 10 times the delay-bandwidth product of the bottleneck link. The RED parameters were maintained as mentioned above, that is the minimum threshold was set to 25% of the buffersize, the maximum threshold to 75% of the buffersize and pmax was set to 0.1. The droprates were in the range 1-4 %. Fig. 5 shows the Fairness Index of DCR-I with respect to TCP-Reno at different values of congestion response delay τ for different buffer sizes.

From the graph we see that the TCP-friendliness is not affected by the amount of buffers available at the bottleneck router. Irrespective of whether the queue can just hold 5 packets or 100 packets at the bottleneck router, for any particular value of τ, the Fairness Index remains almost the same.

3. Fairness Index with a mixed workload

The aim of this simulation is to investigate whether the TCP-friendliness of DCR-I is affected workload mix. The network topology is similar to the experiment investigating the effects of different droprates. However the total number of flows is fixed while

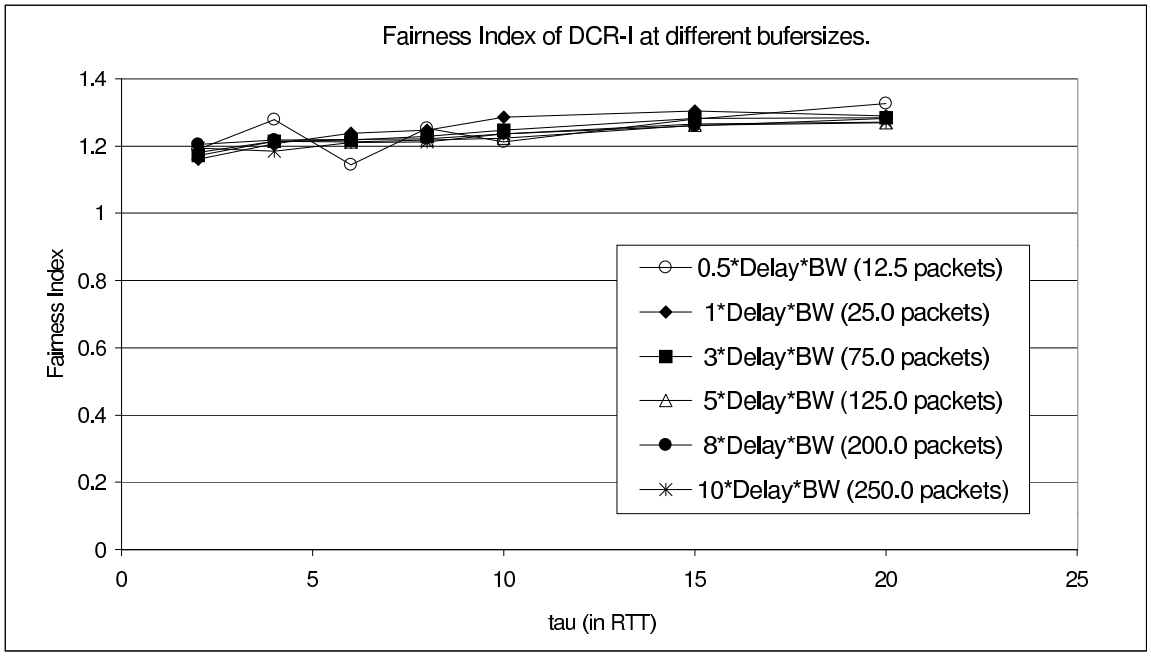


Fig. 5. DCR-I: Fairness Index at Different Buffer Sizes

the percentage of TCP-Reno and DCR-I flows is varied. Fig. 6 shows the results.

From the graph we see that there are a lot of variations in the Fairness Index as the percentage of DCR-I and TCP-Reno flows are varied inspite of keeping the total number of flows constant. With larger number of DCR-I flows in the system, the bandwidth is shared quite fairly. However, when there are lesser DCR-I flows in the system, they manage to claim much more than their fair share of their bandwidth by forcing the TCP-Reno flows to have a larger droprate. These results indicate that even in the presence of RED routers, which inherently balance the per-flow droprates by randomly dropping packets in the presence of congestion, different values of droprates can be seen by different flows, when workload mix is varied. Sample values of the per-flow droprates are provided in Table I.

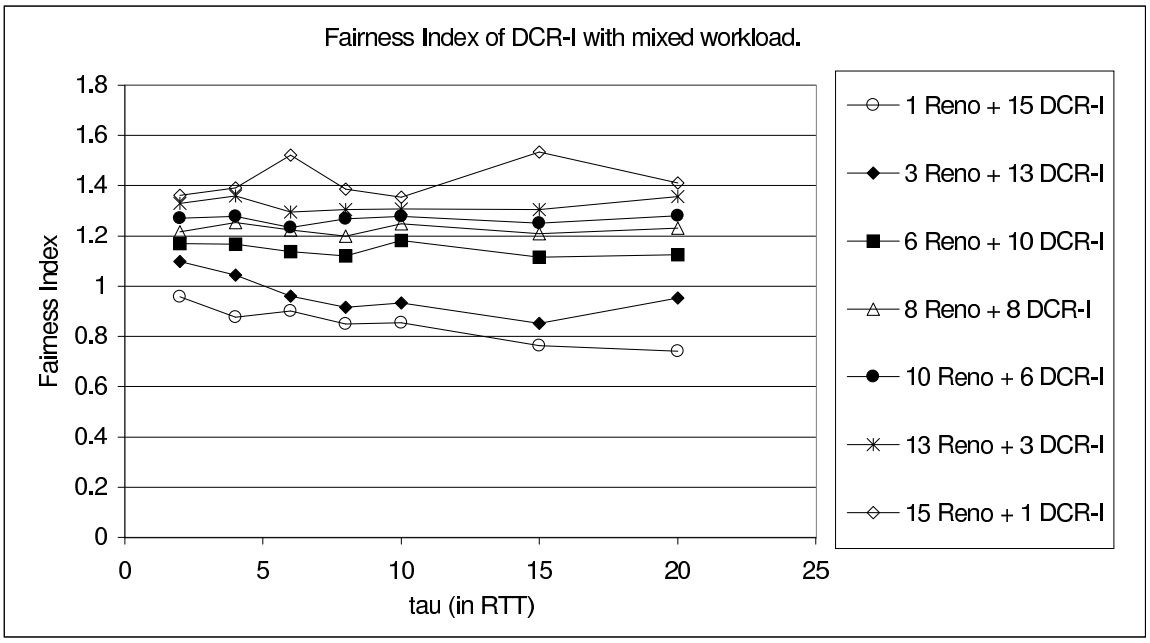


Fig. 6. DCR-I: Fairness Index with Mixed Workload

Table I. DCR-I: Sample Values of Per-flow Droprates for Mixed Load

	1 Reno + 15 DCR-I		**8 Reno + 8 DCR-I**		**15 Reno + 1 DCR-I**	
tao	reno perflow droprate (%)	DCR-I perflow droprate (%)	reno perflow droprate (%)	DCR-I perflow droprate (%)	reno perflow droprate (%)	DCR-I perflow droprate (%)
2	1.669	2.381	1.755	1.719	1.632	1.330
4	1.662	3.060	1.739	1.667	1.607	1.256
6	1.719	3.101	1.711	1.676	1.618	1.067
8	1.608	3.294	1.787	1.707	1.657	1.209
10	1.646	3.391	1.886	1.699	1.642	1.343
15	1.365	3.300	1.749	1.729	1.639	1.063
20	1.253	3.117	1.849	1.755	1.630	1.288

4. Simulations with droptail router

In this simulation, we investigate the TCP-friendliness of DCR-I when the buffer management scheme in the network is droptail. Thus, R1 and R2 were configured to be droptail routers. The rest of the topology is same as the experiment with RED router where we investigate the effects of different droprates on TCP-friendliness of DCR-I.

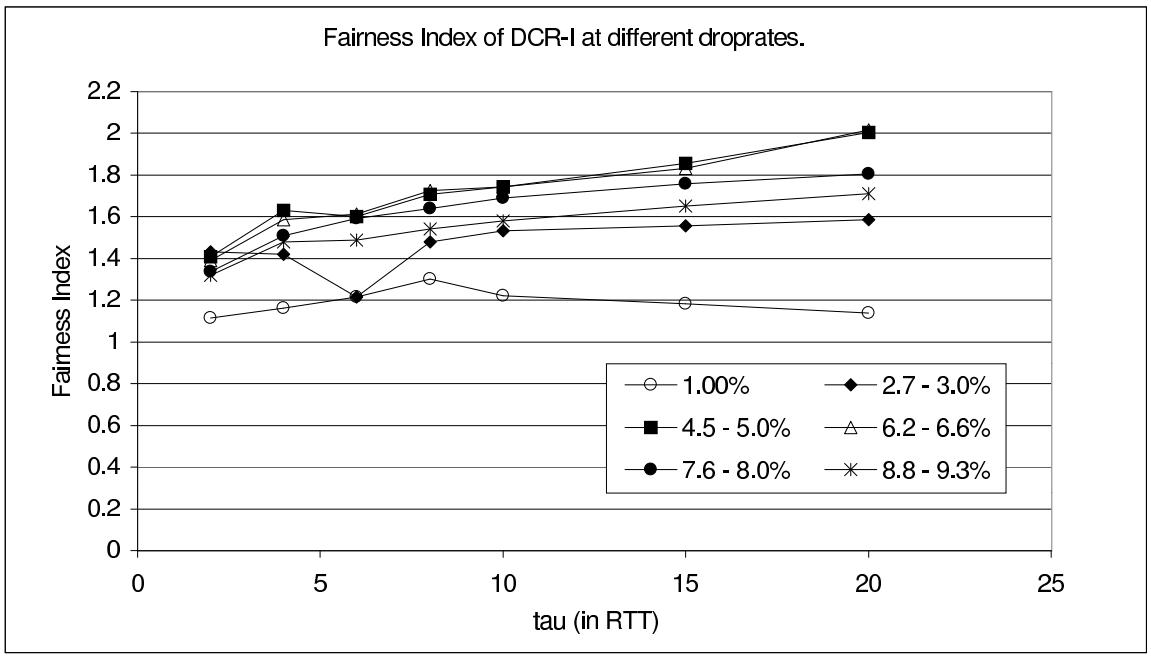


Fig. 7. DCR-I: Fairness Index at Different Droprates with Droptail Router

It may be noticed from Fig. 7 that at higher drop probabilities and for larger values of τ DCR-I manages to get almost two and half times the bandwidth share compared to that of TCP-Reno. Initially, we found these results puzzling, since the steady state analysis shows that the protocol DCR-I should be TCP-friendly. On a closer look at the per-flow droprates, we realised that the congestion response mechanism of DCR-I forces the competing TCP-Reno flow to have a higher droprate, thus inval-

idating the basic assumption of the steady state analysis, that all the flows should have the same drop probability. This result is in agreement with the observation made in [8]. It also supports our intuition that, in case of passive queue management schemes, the delayed response protocols should atleast start to decrease their sending rate when the congestion is perceived. Sample values of perflow droprates are shown in Table II.

Table II. DCR-I: Sample Values of Per-flow Droprates with Droptail Router

Bottleneck Link Droprate : 4.5 - 5.0%			
tao	reno perflow droprate	dcr-i perflow droprate	link droprate
2	5.013	4.286	4.586
4	5.793	4.099	4.748
6	5.639	4.206	4.758
8	5.807	4.142	4.759
10	5.867	4.147	4.774
15	6.127	4.181	4.867
20	6.487	4.214	4.977

5. Fairness Index when number of flows in the network exceed the available buffersize at the bottleneck router

In this simulation, the number of flows in the network in increased while keeping the buffersize at the bottleneck router, comparatively small. The topology consists of bottleneck link with bandwidth 80Mbps and the RTT is set to 22ms. The bottleneck link buffersize is set to 3*delay*bandwidth of the bottleneck link, which is 30 packets. The RED parameters have been set as explained before. The number of flows is

then increased from 30 to 60 and then 90, with half the flows being TCP-Reno and the other half being DCR-I. The link droprate is in the range 1-5 %. The graph corresponding to the results is shown in Fig. 8.

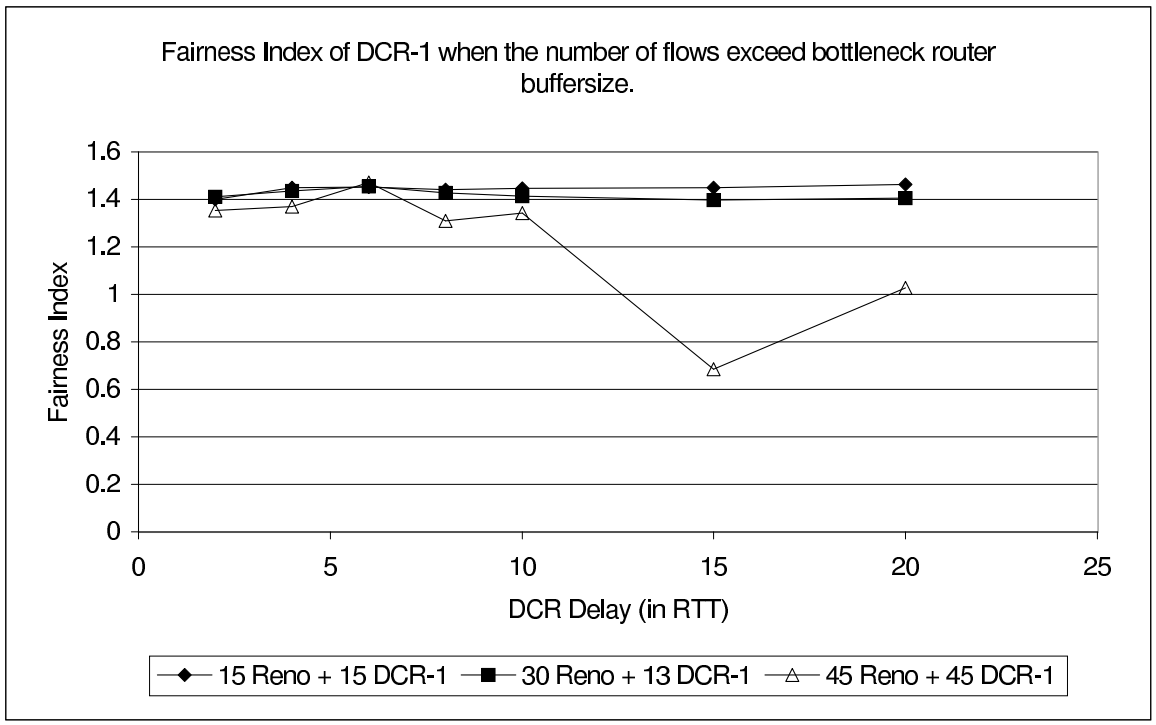


Fig. 8. DCR-I: Fairness Index with More Flows in the Network Than Buffers

It may be noted from the graph that DCR-I has an almost constant Fairness Index of about 1.4 until the τ value of 10 RTTs. For higher values of τ, some of the DCR-I flows were floored and hence the Fairness Index drops drastically. This is more a result of overutilization than the DCR-I algorithm itself.

6. Effect of RED parameters on DCR-I

The simulations with droptail router indicate that unless an early notification is provided to the DCR-I protocol regarding the congestion in the network and unless

there are enough buffers at the bottleneck router to damp the effects of delayed congestion response, it ceases to be TCP-friendly. Since the TCP-friendliness seems to depend on the buffer management scheme we wanted to test the effects of RED parameters on the Fairness Index. We conducted several simulations for this. In Fig. 9 we present the results of two sets of simulations, one in which the maximum threshold was set to 75% of the buffersize and the minimum threshold was varied and in the second set, we set the maximum threshold to 100% and varied the minimum threshold.

It may be noted that the Fairness Index is quite stable unless in the cases where maximum threshold as well as the minimum threshold are close to 100%. This maybe expected, as under these circumstances RED behaves almost as a droptail router. Also it maybe noted that performance can only get better when the parameter maximum threshold is set to smaller values as larger amounts of the buffers will be available to dampen the effects of delayed response.

7. Effects of DCR-I clock resolution

One other thing of interest is the clock resolution used by the DCR-I Implementation. It may be noted that with a lower resolution clock, there will be errors in computing the RTT and this affects the congestion response interval $\tau * RTT$. For a topology with low RTT in the order of 10ms and the clock resolution is of the order 100ms quite a bit of error could be introduced. This is to be expected, as the time the DCR-I protocol spends claiming the bandwidth increases, and so does its bandwidth share. The effects are shown in this simulation. The network topology consisted of 32 flows, bottleneck link bandwidth of 10Mbps, RTT of 26 ms and buffer size of 3*delay*bandwidth of the bottleneck link. The same simulation was conducted once with a 100ms clock resolution and once with a 10ms clock resolution. The results are

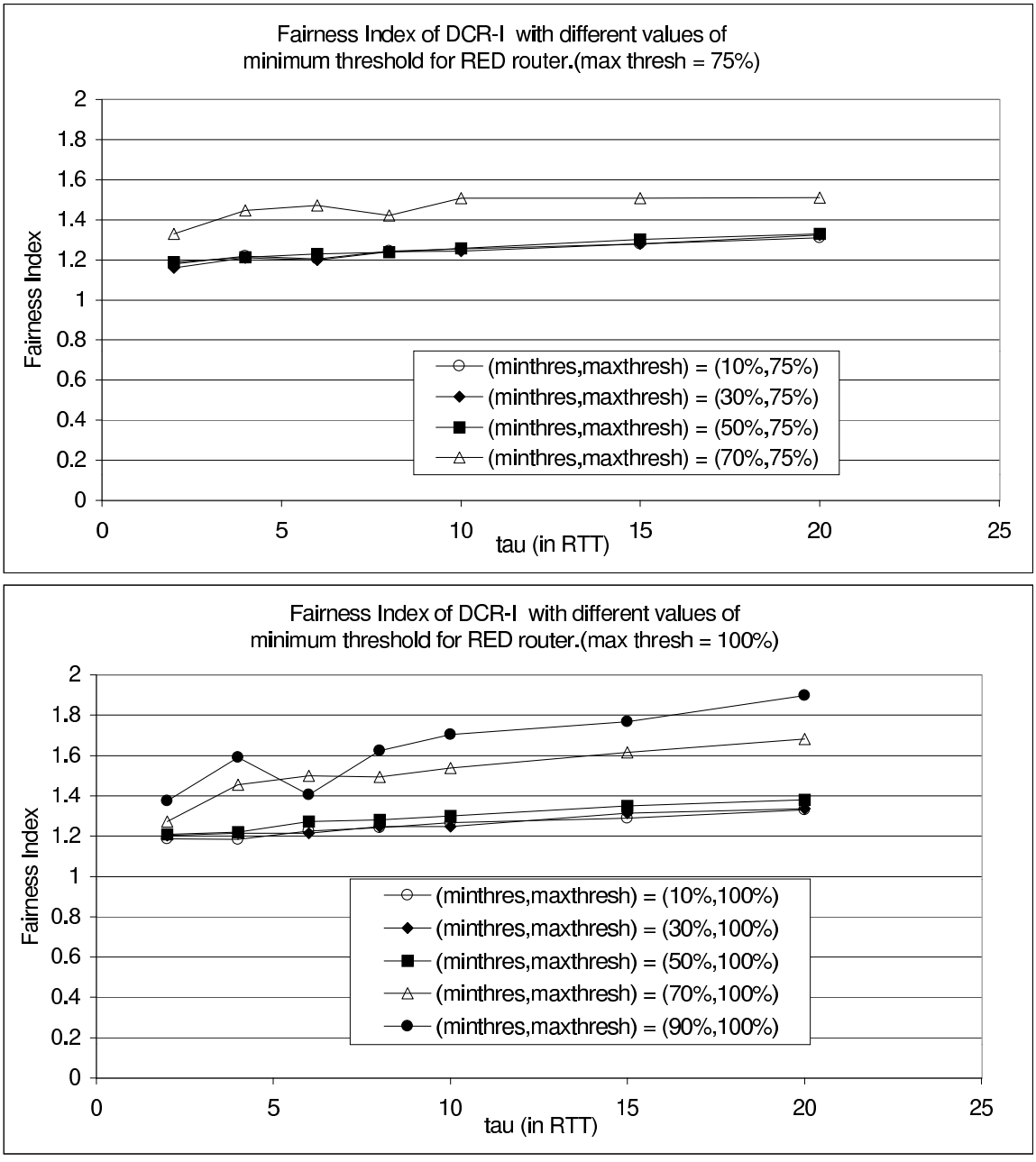


Fig. 9. DCR-I: Effect of RED Parameters

shown in Fig. 10.

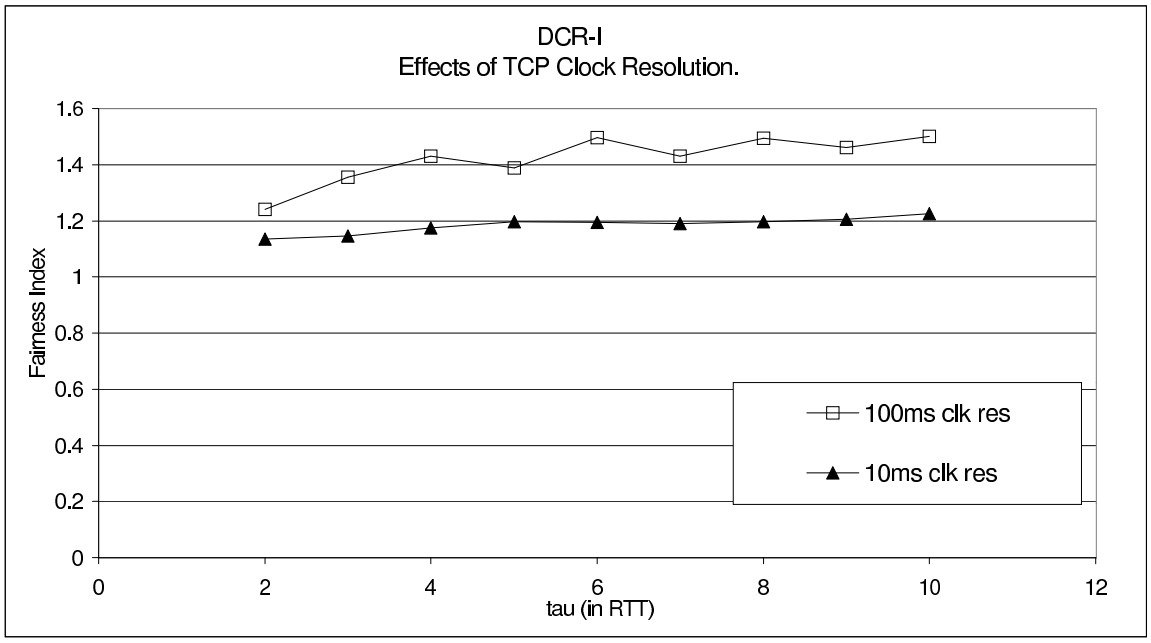


Fig. 10. DCR-I: Effect of Clock Resolution

It may be noted that with a higher resolution clock a much better performance is achieved. As explained in the section on implementation, DCR-I was implemented based on the existing TCP-Reno implementation and all the simulation results shown are with the default clock resolution of 100ms. As a result, the results shown stray away from the analysis results due to the engineering issues in the clock implementation. Since our goal is to verify the analysis using simulations, we intend not to get bogged down with the engineering problems, and so will be repeating simulations with a 10ms clock resolution. Until those results are included here, we request the reader to judge the results presented here in a liberal manner.

8. Simulations with different set of values for DCR-I parameters.

As indicated in equation (3.9) a wide range of values can be chosen for the parameters α and γ. All the simulations presented above were conducted with the value of $\alpha = 1$ and $\gamma = 0.5$. In this section we present the simulation results for a different set of values for the DCR-I parameters. Here γ was set to 0.8 and α to 0.333. The rest of the simulation topology was exactly the same as that for the simulation investigating the effects of droprates on Fairness Index. Fig. 11 shows the results.

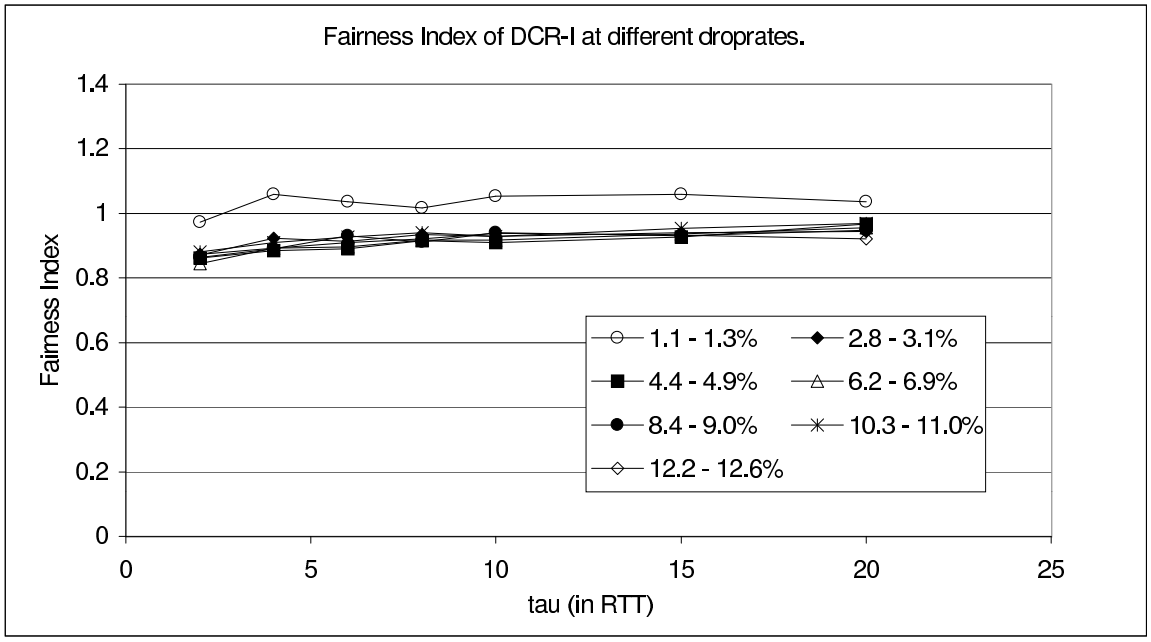


Fig. 11. DCR-I: Fairness Index for Different Values of DCR-I Parameters

As may be noted from the graph the Fairness Index is within 20% of the ideal value of 1. It may also be noted that for any particular value of the droprate, the Fairness Index is almost independent of the droprate except at low droprates.

9. Smoothness of DCR-I protocol

One of the primary requirements for real-time multimedia applications is that variations in the sending rate should be minimum. We conducted some preliminary tests to verify the smoothness of the DCR-I protocol. The results are presented in this section.

To quantize and understand the smoothness of the DCR-I protocol we use the measure, coefficient of variance, which is defined as the standard deviation divided by the mean. This approach has been used before to verify the smoothness in [6]. We have used a similar approach as explained in the above mentioned paper. The throughput of each flow is noted over small units of time indicated by δ and the Coefficient of Variance for these values is calculated.

In the first experiment we took the throughput readings every 0.5 seconds and computed the co-efficient of variance for different values of the droprates. The number of flows in the topology was fixed at 16 and the bandwidth was varied to obtain the different droprates. The buffer size was set to 3*delay*bandwidth of the bottleneck link. An RTT of 100ms was used and the clock resolution was 10ms. Coefficient of variance was computed for each flow. These coefficients of variances were then averaged for each type of flow. Table III shows the results for the two sets of simulations, one with $\alpha = 1.0$, γ =0.5 and the other with $\alpha = 0.333$, $\gamma = 0.8$. As may be expected, the coefficient of variance for DCR-I with $\alpha = 1$ and $\gamma = 0.5$ is larger than that of TCP-Reno. However, by choosing the parameters differently, DCR-I may be made to have a smoother sending rate as indicated by the results for $\alpha = 0.333$, $\gamma = 0.8$.

In the second experiment we kept the droprate fixed and varied the value of δ. Again 16 flows were used. The bandwidth of the bottleneck link was fixed at 50Mbps. Rest of the parameters were the same as the previous experiment. A droprate of 0.1%

Table III. DCR-I: Coefficient of Variance at Different Droprates

Bottleneck Link Droprate (%)	TCP-reno	DCR-I (alpha = 1.0 gamma = 0.5)
14.85	0.775	0.869
9.93	0.841	0.904
8.75	0.710	0.753
7.34	0.627	0.664
6.35	0.580	0.616
5.38	0.550	0.570
4.21	0.513	0.523
3.32	0.484	0.480
2.75	0.465	0.460
2.3	0.448	0.443

Bottleneck Link Droprate (%)	TCP-reno	DCR-I (alpha = 0.333 gamma = 0.8)
12.7	0.797	0.807
8.7	0.888	0.756
7.5	0.704	0.635
6.4	0.605	0.547
5.4	0.537	0.493
4.5	0.499	0.467
3.5	0.484	0.434
2.7	0.450	0.403
2.2	0.445	0.368
1.9	0.428	0.354

was observed at the bottleneck link. The results as shown in Table IV indicate that DCR-I has lesser variance than TCP-Reno.

Table IV. DCR-I: Coefficient of Variance at Different Values of δ

delta (in seconds)	TCP-reno	DCR-I (alpha = 1.0 gamma = 0.5)
0.1	0.405	0.377
0.5	0.347	0.279
1	0.323	0.252
2	0.280	0.218
5	0.218	0.174
10	0.170	0.137
15	0.145	0.114

delta (in seconds)	TCP-reno	DCR-I (alpha = 0.333 gamma = 0.8)
0.1	0.373	0.240
0.5	0.318	0.155
1	0.300	0.143
2	0.255	0.136
5	0.206	0.126
10	0.154	0.114
15	0.129	0.104

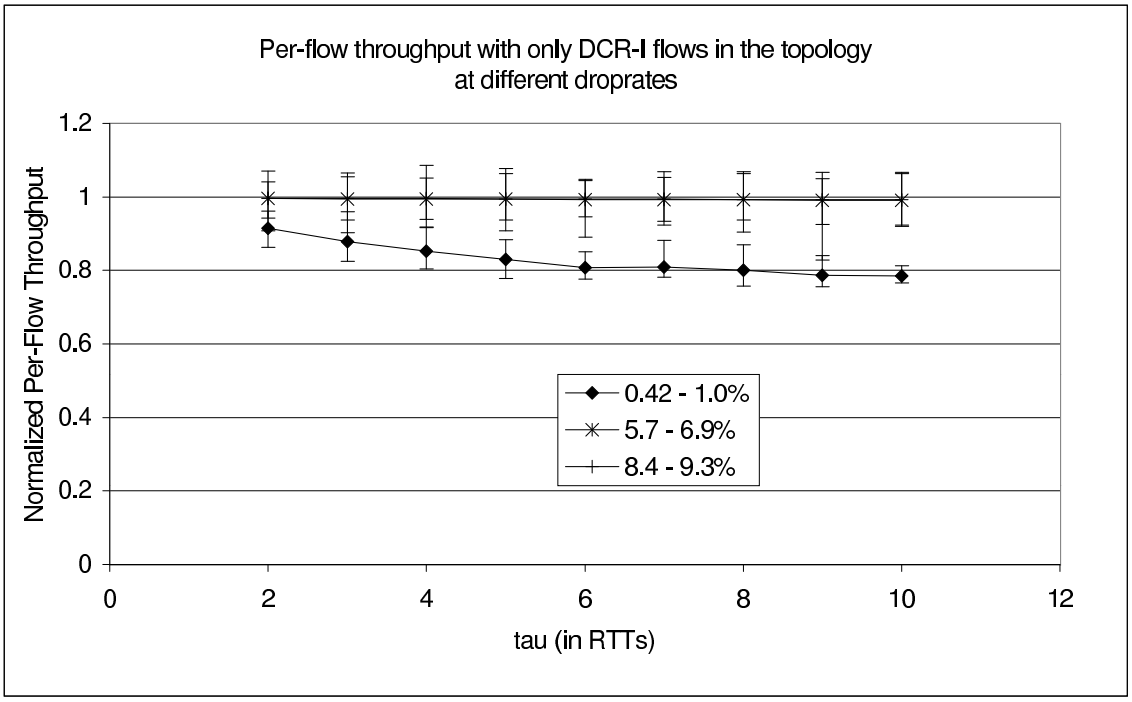
From the results it is clear that DCR-I protocols have much smoother sending rate than TCP-Reno when compared over small time intervals. The smoothness of the protocols can be varied according to the needs of the application by simply adjusting the DCR-I parameters α and γ.

10. Simulations with only DCR-I flows

In order ot investigate the effects of having only DCR-I flows in a network, we conducted the following set of simulations. The bottleneck link bandwidth was set to 10Mbps and the end-to-end round trip time was set to 100ms. The number of flows in the network was varied to obtain different droprates. Fig. 12 shows the results. In the first graph the normalized throughput of each flow is plotted with the solid line indicating the average normalized per-flow throughput. In the second graph we show the link utilization at the bottleneck link. We notice from the graph that except at very low droprates, when the parameter τ is set very high, the bandwidth is utilized well and shared in a fair manner.

D. Conclusions

From the steady state analysis we have proved that Delayed congestion response protocols with the values of $f_1(t)$ and $f_2(t)$ as indicated, which continue to increase the sending rate upon congestion notification can be TCP-friendly. Through simulations, we have shown that they are TCP-compatible under a wide range of network parameters in the presence of active queue management schemes like RED. The results seem to indicate that in order to be TCP-friendly even in the absence of active queue management schemes, the function $f_2(t)$ should be a decreasing function. We investigate the affects of such a function in the next chapter.



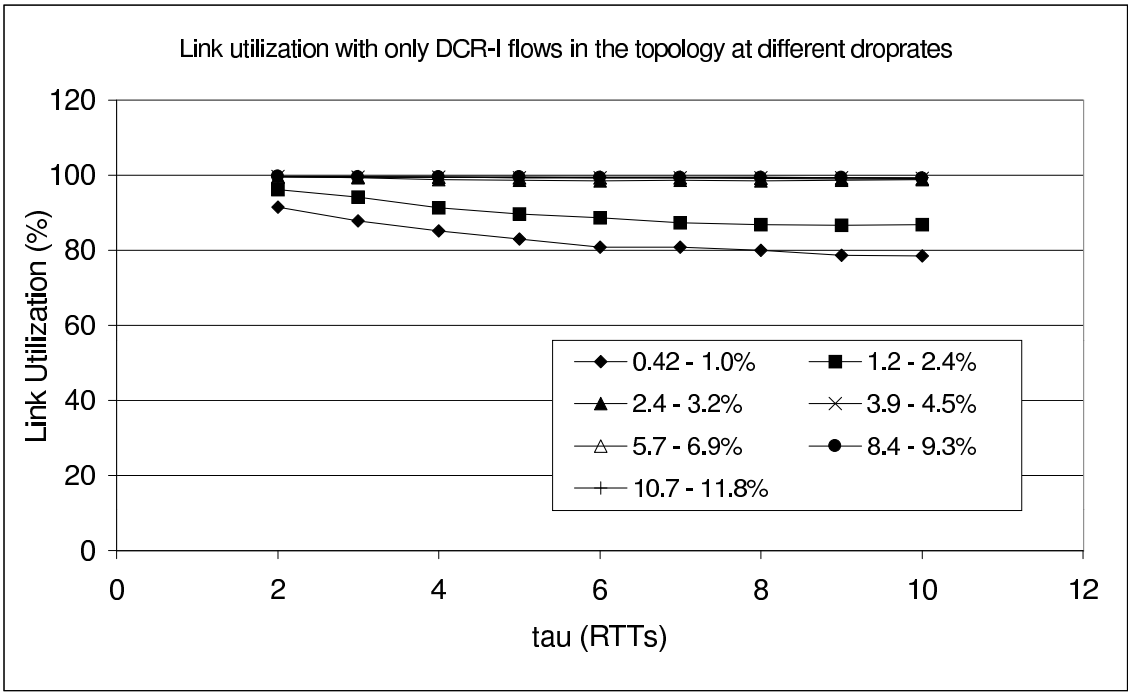


Fig. 12. DCR-I: Simulation Results with Only DCR-I Flows in the Network

CHAPTER IV

DCR-D: USING A DECREASING FUNCTION FOR $f_2(t)$

In this algorithm, the congestion window is decreased gradually over a period of τ RTTs on congestion notification. Since $f_2(t)$ is already a decreasing function, we can set γ to 1. We tried several functions for $f_2(t)$ to verify TCP-friendliness. The simplest one which satisfies the TCP-friendliness requirement is presented here. The control equations can be written as follows:

$$\begin{aligned} f_1(t) \quad : w_{t+R} &\leftarrow w_t + \alpha; \quad \alpha > 0 \\ f_2(t) \quad : w_{t+R} &\leftarrow \beta * w_t; \quad 0 < \beta < 1, t_{drop} < t \leq t_{drop} + \tau \\ w_{tdrop+\tau} &\leftarrow \gamma * w_{tdrop+\tau-\epsilon}; \quad \gamma \leq 1 \end{aligned} \tag{4.1}$$

A. Simple Analysis

In this section we present some details of the analysis of this protocol. The analysis is done along the similar lines of DCR-I. In order to simplify the analysis we set the parameter γ to 1. The average throughput λ can be computed as the number of packets sent between two successive drops divided by the time between two successive drops. The number of packets sent between two successive drops is 1/p. Hence we have

$$\lambda = \frac{\frac{1}{p}}{(t_2 - t_1) + \tau R} \tag{4.2}$$

To find $(t_2 - t_1)$:

Since $f_1(t)$ is the same for DCR-I and DCR-D, the equation for $(t_2 - t_1)$ remains

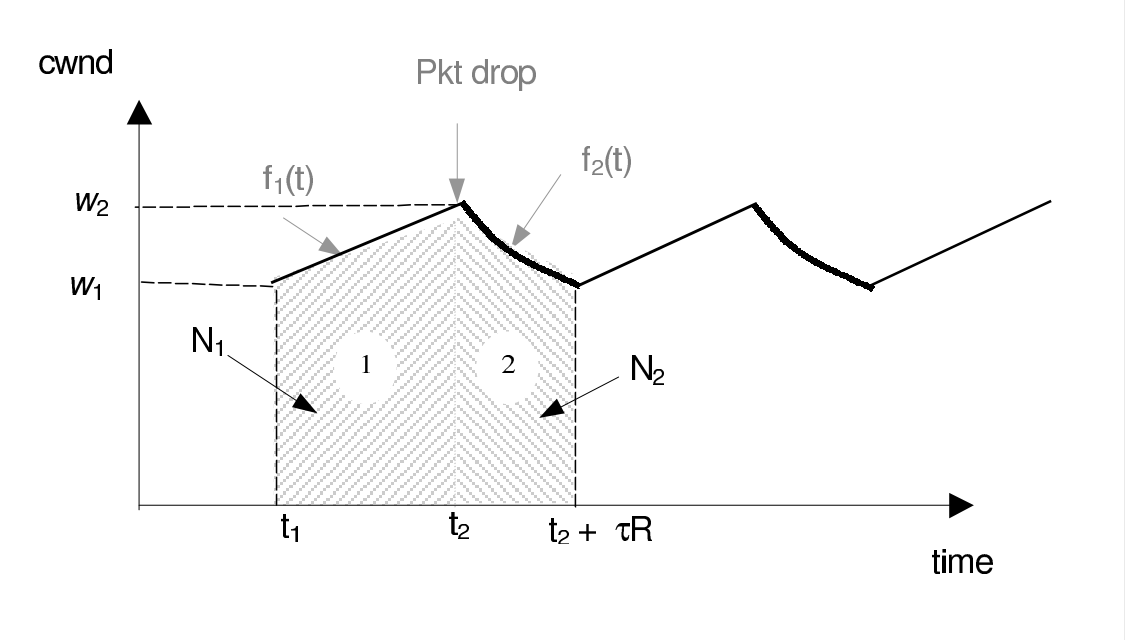


Fig. 13. Congestion Window for DCR-D Protocol

the same as in equation (3.4). Thus we have -

$$t_2 - t_1 = \frac{R}{\alpha}(w_2 - w_1) \tag{4.3}$$

From Fig. 13, we see that, when the packet drop occurs the congestion window is w_2. For τ RTTs after the congestion notification, the window is reduced at a rate of β times the current window for each RTT. Thus at the end of τ RTTs, the size of the congestion window will be $\beta^\tau w_2$. This is nothing but w_1. Thus we have

$$w_1 = \beta^\tau w_2 \tag{4.4}$$

Substituting (4.4) in equation (4.3) we have

$$t_2 - t_1 = \frac{R}{\alpha} w_2 (1 - \beta^\tau) \tag{4.5}$$

So now the throughput equation as given in (4.2) may be written as

$$\lambda = \frac{1}{\frac{Rp}{\alpha}\left(1-\beta^{\tau}\right)w_2 + \tau Rp}$$

The term β^{τ} is a constant. In order to simplify the equations, we replace it with K. Thus we have

$$\lambda = \frac{1}{\frac{Rp}{\alpha}(1-K)w_2 + \tau Rp} \tag{4.6}$$

To find the value of w_2 :

We know that the total number of packets sent between two successive drops is 1/p. This is nothing but the shaded region of the graph. Thus

$$N_1 + N_2 = \frac{1}{p} \tag{4.7}$$

The function $f_1(t)$ is the same in DCR-D as in DCR-I. So from (3.3) we have,

$$\begin{aligned} N_1 &= \int_{t_1}^{t_2} f_1(t)dt \\ &= \int_{t_1}^{t_2} \frac{\alpha t}{R} \cdot \frac{dt}{R} \\ &= \frac{\alpha}{R^2}\left(\frac{t_2^2}{2} - \frac{t_1^2}{2}\right) \\ &= \frac{1}{2\alpha} w_2^2 \left(1-\beta^{2\tau}\right) \end{aligned} \tag{4.8}$$

The function $f_2(t)$ may be written in terms of the widow size w_2 as follows

$$w(t) = w_2.\beta^t$$

Using this we can compute the number of packets N_2 sent in the shaded region (2) as follows

$$N_2 = \int_{t_2}^{t_2+\tau} f_2(t)dt$$

$$
\begin{aligned}
&= \int_0^{\tau} w_2 \beta^t dt \\
&= \left[w_2 \frac{\beta^t}{ln\beta} \right]_0^{\tau} \\
&= \frac{w_2}{ln\beta} (\beta^{\tau} - 1)
\end{aligned} \tag{4.9}
$$

Substituting these values in equation (4.7) we have

$$
\frac{1}{2\alpha} w_2^2 \left(1 - \beta^{2\tau}\right) + \frac{w_2}{ln\beta} (\beta^{\tau} - 1) = \frac{1}{p} \tag{4.10}
$$

As explained above, β^{τ} is a constant and is replaced by K. This equation is now a quadratic equation with its roots giving the value of w_2 in terms of α, β, τ and p. Thus after simplification, we have

$$
w_2 = \frac{2\alpha + \sqrt{4\alpha^2 + 8 \cdot ln^2\beta \cdot \frac{\alpha(1+K)}{p(1-K)}}}{2 \cdot ln\beta \cdot (1+K)} \tag{4.11}
$$

We simplify the above equation at the risk of overestimating the value of w_2 by making the assumption that $\sqrt{A^2 + B^2} = A + B$ for the numerator. Thus we have,

$$
w_2 = \frac{2\alpha + \sqrt{\frac{2\alpha(1+K)}{p(1-K)}} \cdot ln\beta}{ln\beta \cdot (1+K)} \tag{4.12}
$$

Inserting this value in the equation (4.6) we get

$$
\begin{aligned}
\lambda &= \frac{1}{\frac{2Rp(1-K)}{ln\beta \cdot (1+K)} + R\sqrt{\frac{2p(1-K)}{\alpha(1+K)}} + \tau Rp} \\
&= \frac{1}{Rp\tau \left(\frac{2}{\tau ln\beta} \cdot \frac{(1-K)}{(1+K)} + 1 \right) + R\sqrt{\frac{2p(1-K)}{\alpha(1+K)}}} \\
&= \frac{1}{R\left(p\tau \left(\frac{2}{lnK} \cdot \frac{(1-K)}{(1+K)} + 1 \right) + \sqrt{\frac{2p(1-K)}{\alpha(1+K)}} \right)}
\end{aligned} \tag{4.13}
$$

The above equation gives the throughput for a DCR-D flow. However, for the DCR-D flow to be TCP-friendly, it should satisfy the condition $\lambda \quad \alpha \quad \frac{1}{\sqrt{p}}$. For that to

happen, the above equation should have only the second term in the denominator. This implies that,

$$\begin{aligned} p\tau\left(\frac{2}{ln\beta}\cdot\frac{(1-K)}{(1+K)}+1\right) &= 0 \\ \frac{2(1-K)}{(1+K)}+lnK &= 0 \end{aligned} \tag{4.14}$$

When K satisfies the above equation we have

$$\lambda = \frac{\sqrt{\frac{\alpha(1+K)}{2(1-K)}}}{R\sqrt{p}} \tag{4.15}$$

Comparing this with the TCP equation we obtain the value for α as

$$\begin{aligned} \frac{3}{2} &= \frac{\alpha(1+K)}{2(1-k)} \\ \alpha &= \frac{3(1-K)}{(1+K)} \end{aligned} \tag{4.16}$$

B. DCR-D Parameters

Now we take a closer look at the values of K and α required to make DCR-D TCP-friendly. First we look at equation (4.14). In order to satisfy this equation, K should be equal to 1. This implies that β should be 1. But for the function $f_2(t)$ to be a decreasing function β should be less than 1. Setting K in the range 0.6 - 1.0 introduces negligibly small amounts of error as indicated in the Table V.

Several simulations were conducted to evaluate the effects of the parameter K on TCP-friendliness of DCR-D protocols. Fig. 14 shows the resulting graphs. It may be noted from these results that *the parameter K primarily dictates the dependence of the Fairness Index on the droprate p and the delay τ.* This is evident from the equation (4.13) which contains an additional term containing p and τ whose effect is felt more and more as the error due to K increases. This error starts to manifest

Table V. DCR-D: Error Introduced by Parameter K

K	$\frac{2}{lnK} \cdot \frac{(1-K)}{(1+K)} + 1$
0.9	0.0009
0.8	0.0041
0.7	0.0105
0.6	0.0212
0.5	0.0382
0.4	0.0646
0.3	0.1055

itself at lower values of K where instead of being bunched up together, the curves are spread out indicating that the Fairness Index is starting to depend on p. Also as K is increased DCR-D flows get much lesser than their fair share for larger values of the delay τ. Thus we concluded that it is favorable for the DCR-D protocol to use values of K as close to 1 as possible. At the same time we note that a value of K very close to 1, indicates that the congestion window is reduced very less on congestion notification. This calls for a trade off in choosing the value of K and we have chosen to set it to 0.8 for all our simulations. With this value of K the equation (4.13) is reduced to

$$\lambda = \frac{1}{R\left(0.0041 * p\tau + \sqrt{\frac{2p(1-K)}{\alpha(1+K)}}\right)} \tag{4.17}$$

The parameter α can be calculated for any chosen value of K using the equation (4.16).

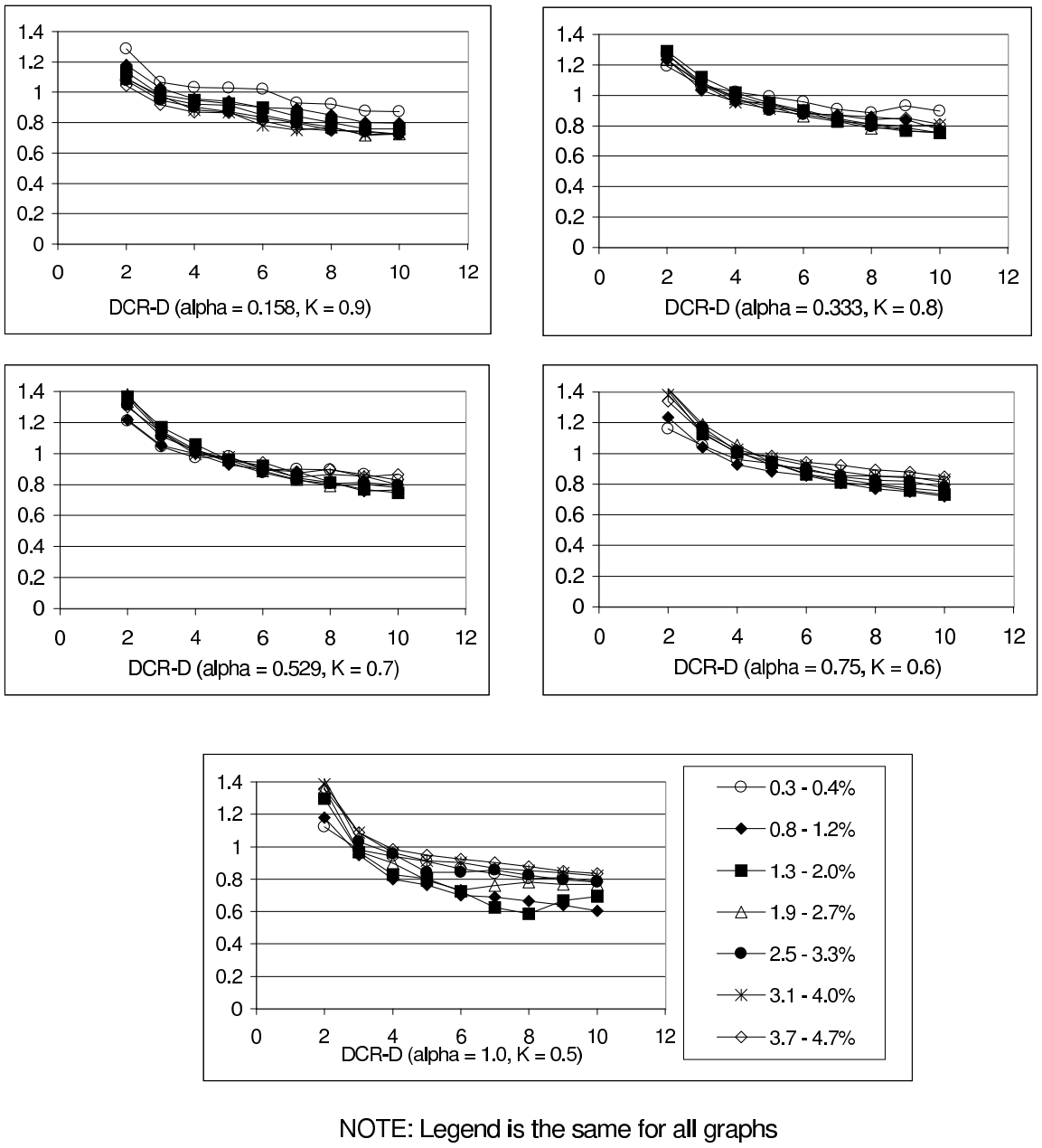


Fig. 14. DCR-D: Fairness Index at Different Values of K

C. Implementation

The DCR-D protocol has two different modes of operation. When there is no congestion in the network, the protocol seeks out bandwidth using the same control equation as TCP as indicated by $f_1(t)$. We call this the *increase* mode. Upon the notification of congestion (indicated by three dupacks), the protocol sets a timer called the *delayed response timer* to expire at the end of τ RTTS and enters the *decrease* mode. During the decrease mode, the congestion window is reduced by the factor $\beta\frac{1}{cwnd}$ on the receipt of each acknowledgement. Thus at the end of each RTT, the congestion window is reduced by the factor β. When the delayed response timer times out, we return to the increase mode.

One other factor to be taken into consideration is how to react to a packet drop in the decrease mode. We reason that if there is a packet drop in the decrease mode, then it is an indication that the network is severely congested and more drastic action is required than the smooth response of DCR-D. So when a packet is dropped during the decrease mode, the congestion window is drastically reduced by setting it to the target value of what it should be at the end of τ RTTs. From the implementation perspective, when the delayed response timer is set, we note the target value of the congestion window at the end of τ RTTs, which is β^τ times the current congestion window. When a packet drop is perceived, if there is a pending delayed response timer, then we reset it, set the congestion window to the target value and set the timer again in response to the latest drop and note the target value for it.

D. Simulation Results

The network topology was exactly the same as in DCR-I. Exceptions will be mentioned where appropriate. The DCR-D parameters were set to $K = 0.8$ and α =0.333

for most of the simulations. If these parameters are changed, then it will be mentioned in the explanation of the concerned simulation.

1. Fairness Index at different buffer sizes

For this experiment, the bandwidth of the bottleneck link was set to 10 Mbps. The end-to-end RTT was set to 100ms. The topology consisted of 32 flows, half of which were TCP-Reno and the other half were DCR-D. The queue length of the bottleneck router was increased from 1 to 10 times the delay-bandwidth product of the bottleneck link. A value of 0.8 was chosen for the DCR-D parameter K and hence α was set to 0.333. The link Utilization was greater than 97% (in most cases greater than 99%) and the link droprate was in the range 2-4.5%. The DCR-D clock resolution was set to 10ms. Fig. 15 shows the resulting graph

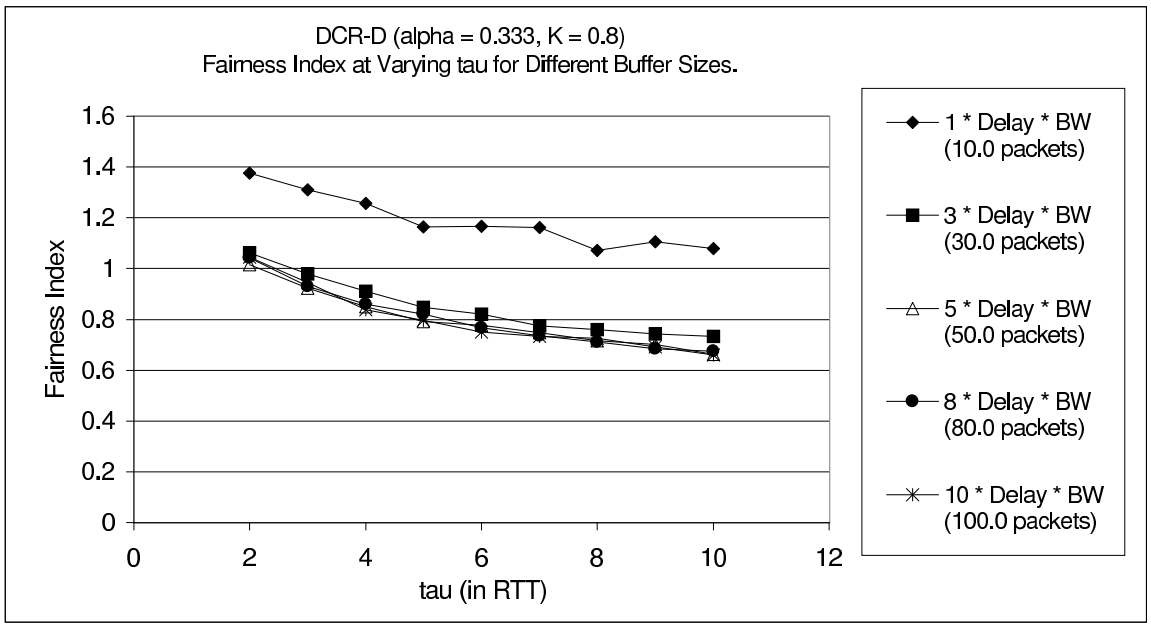


Fig. 15. DCR-D: Fairness Index at Different Buffer Sizes

From the graph we see that for any particular value of τ the amount of buffers at

the bottleneck router does not affect the Fairness Index, when a reasonable amount of buffers is available. However when the number of buffers is low compared to the number of number of flows in the network, then the DCR-D flows tend to hog the bandwidth. This is evident in the curve for 1*delay*BW, where the queue length is 10 packets where as there are 32 flows in the topology. With sufficient buffers available the Fairness Index for DCR-D is within the range of 20% of the ideal value of 1 for τ values of upto 6 RTTs. If τ is set to a greater value DCR-D gets much less than its fair share. This is because the value of K is not 1, as required by the condition in equation (4.14). Due to the requirements of the application, if the DCR-D has to be operated at a larger τ value then the error introduced can be compensated by choosing a slightly higher value of α.

2. Fairness Index at different droprates

In this experiment we aim to investigate exactly how much the drop rate p effects the Fairness Index of DCR-D for a selected value of K and α. The bandwidth of the bottleneck link is 10MBps and the RTT is 100ms. The buffer size of the bottleneck router is set to 6*delay*Bandwidth of the bottleneck link. The number of flows was increased from 16 to 106 to obtain the different droprates. All the while, 50% of the flows were TCP-Reno and the other 50% of the flows were DCR-D. Fig. 16 shows the results. It may be noted from the graph that the Fairness Index depends on the droprate p only for small valuers of τ.

3. Fairness Index with mixed workload

In most of the simulations the topology was setup such that half of the flows were of TCP-Reno and the other half were DCR-D. In this simulation we investigate the effects of having mixed workload. The bandwidth of the bottleneck link was set as

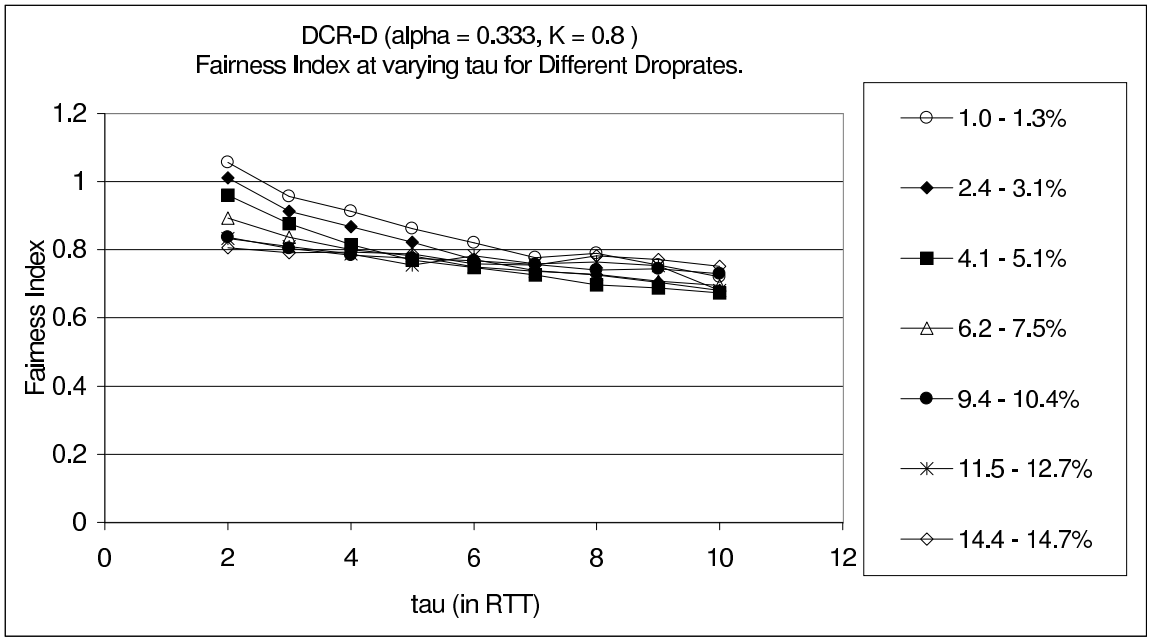


Fig. 16. DCR-D: Fairness Index at Different Droprates

10Mbps and the RTT was 100ms. The DCR-D parameters were as mentioned before, that is, $K = 0.8$ and $\alpha = 0.333$. The total number of flows was fixed as 16, however the percentage of flows belonging to each protocol was varied. The results of the simulation is presented in Fig. 17.

From the graph we see that the result is quite counter-intuitive. With larger number of DCR-D flows in the system, the bandwidth is shared quite fairly, with TCP-Reno getting larger share for larger values of τ. However, when there is just one or three DCR-D flows in the system, they manage to claim much more than their fair share of their bandwidth by forcing the TCP-Reno flows to have a larger droprate. This is a very puzzling result and shows that TCP-friendliness is quite a complicated concept. Sample values of the per-flow droprates are provided in Table VI.

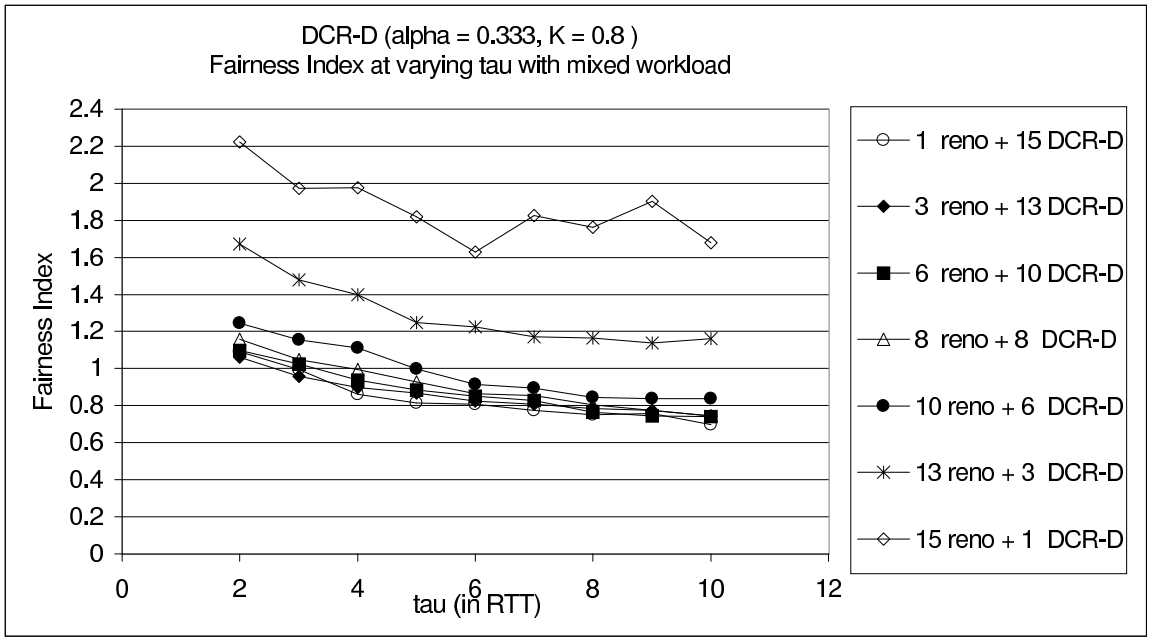


Fig. 17. DCR-D: Fairness Index with Mixed Workloads

Table VI. DCR-D: Sample Values of Per-flow Droprate for Mixed Load

tau	1 reno + 15 DCR-D		8 reno + 8 DCR-D		15 reno + 1 DCR-D	
	TCP-reno perflow droprate (%)	DCR-D perflow droprate (%)	TCP-reno perflow droprate (%)	DCR-D perflow droprate (%)	TCP-reno perflow droprate (%)	DCR-D perflow droprate (%)
2	1.683	1.571	1.641	1.373	1.733	0.550
3	1.491	1.419	1.527	1.307	1.693	0.582
4	1.234	1.332	1.459	1.240	1.696	0.533
5	1.150	1.270	1.387	1.206	1.679	0.556
6	1.139	1.195	1.308	1.183	1.652	0.615
7	1.072	1.153	1.300	1.136	1.675	0.493
8	1.040	1.097	1.239	1.106	1.669	0.485
9	1.030	1.055	1.211	1.108	1.691	0.425
10	0.919	1.011	1.180	1.068	1.661	0.473

4. Simulations with droptail router

As it has been explained before, since the congestion control algorithms used in delayed response protocols are different from that in TCP-Reno, the protocols are not strictly TCP-friendly in the absence of buffer management schemes that explicitly force the different flows to have same droprates. This observation was made in [8] and we found that our simulations gave similar results. However, by setting the parameters of the DCR protocols to be similar to that of TCP-Reno, we can achieve reasonable performance. I order to show this we conducted an experiment by setting the DCR-D parameters to $(\alpha,K) = (1.0,\ 0.5)$. Droptail router was used at the bottleneck link. The results are presented in Fig. 18. It may be noticed from this graph that we can use Delayed Response Protocols, even in the presence of droptail router. However, the value of the parameters need to be chosen as mentioned above. By conducting simulations for these set of parameter values we noted that the smoothness is the same as TCP-Reno or in some cases worse than TCP-Reno. This calls for a trade off. If Delayed Response Protocols need to be used with droptail routers, then the applications should be built to take advantage of early warning regarding the reduction in sending rate and take appropriate measures to provide smooth sending rates.

Table VII shows the sample values of the per-flow droprates. For larger values of τ, the average per-flow droprates observed by DCR-D flows is lesser than the TCP-Reno flows. As a result, for these values of τ, the Fairness Index goes higher than 1.

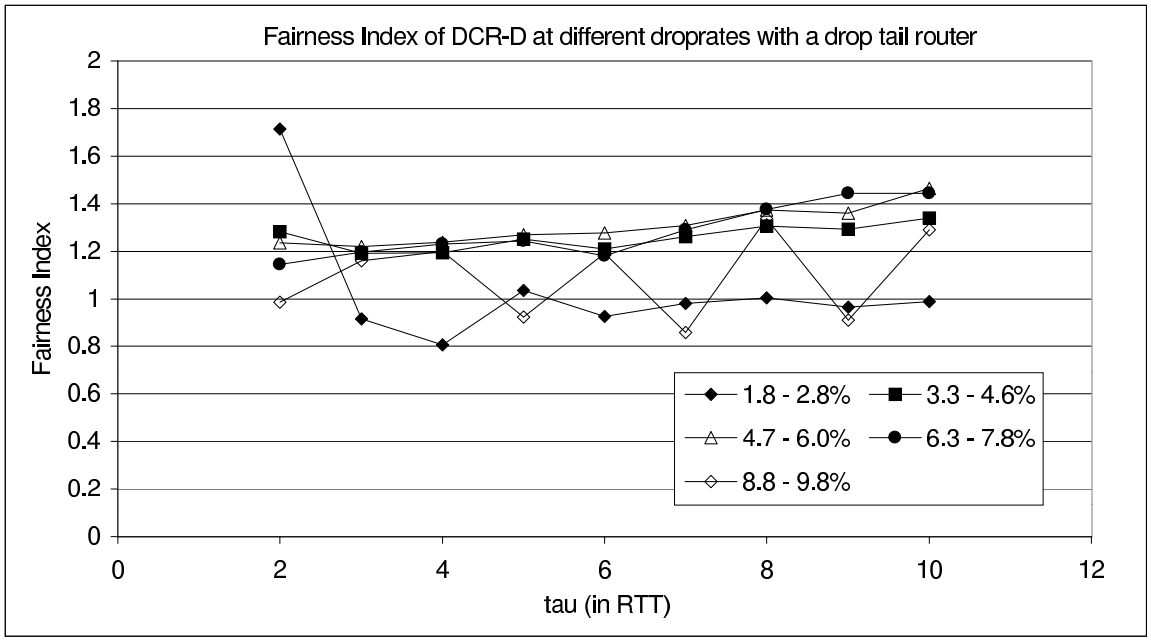


Fig. 18. DCR-D: Fairness Index at Different Droprates with Droptail Router

Table VII. DCR-D: Sample Values of Per-flow Droprates with Droptail Router

Bottleneck Link Droprate : 3.3 - 4.6%			
tao	reno perflow droprate	dcr-d perflow droprate	link droprate
2	5.169	4.099	4.551
3	4.599	3.891	4.206
4	4.635	3.415	3.966
5	4.826	2.949	3.783
6	4.667	2.799	3.642
7	4.800	2.527	3.536
8	4.989	2.283	3.466
9	5.006	2.111	3.388
10	5.112	1.993	3.340

5. Effects of DCR-D clock resolution

One other thing of interest is the clock resolution used by DCR-D while computing the RTT and scheduling the congestion response delay timer. It may be noted that with a lower resolution clock, several errors will be introduced. First of all, if the delayed response timer expires just after the previous clock tick, then the end of delayed response will not be scheduled until the next clock tick. Secondly, the RTT calculation itself will have a certain amount of error and since the timer is scheduled to expire at the end of τ * RTT seconds, additional error will be introduced. As a result the DCR-D algorithm spends longer time in the *decrease mode*. Thus the throughput share obtained by DCR-D decreases. In order to investigate this effect we conducted the following simulation. The network topology consisted of 32 flows, bottleneck link bandwidth of 10Mbps, RTT of 26 ms and buffer size of 3*delay*bandwidth of the bottleneck link. The same simulation was conducted once with a 100ms clock resolution and once with a 10ms clock resolution. The results are shown in Fig. 19.

It may be noted from the graph that the curve for the simulation with 100ms clock resolution lags the curve for the simulation with 10ms clock resolution. However, with DCR-D there is an easy way around this problem. If it is known ahead of time that the clock has a low resolution, then the parameter α can be set to a larger value to compensate for the effects of clock resolution. In order to show this, we repeated the above mentioned experiment with an α value three times that indicated by the equation, that is α was set to 1.0. The results are shown in Fig. 20.

It may be noted that the performance of with 100ms clock resolution is indeed improved. Also, to be noted is that increasing the α value does not drastically increase the performance at 10ms clock resolution, whose Fairness Index is still within the acceptable range of 20% from the ideal value of 1.

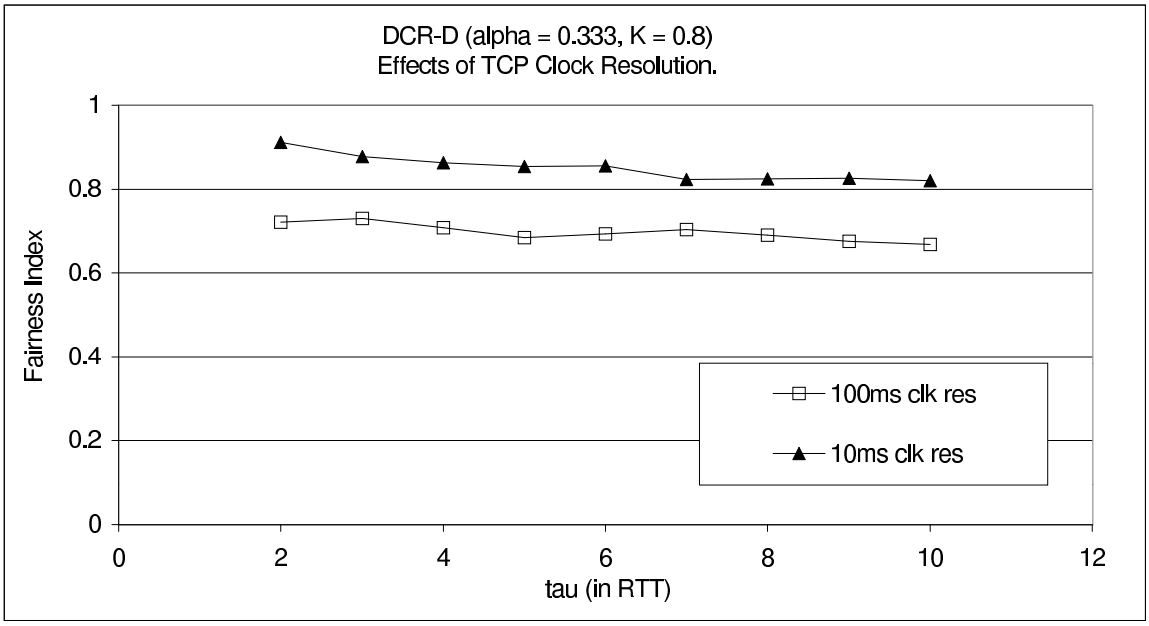


Fig. 19. DCR-D: Effects of Clock Resolution

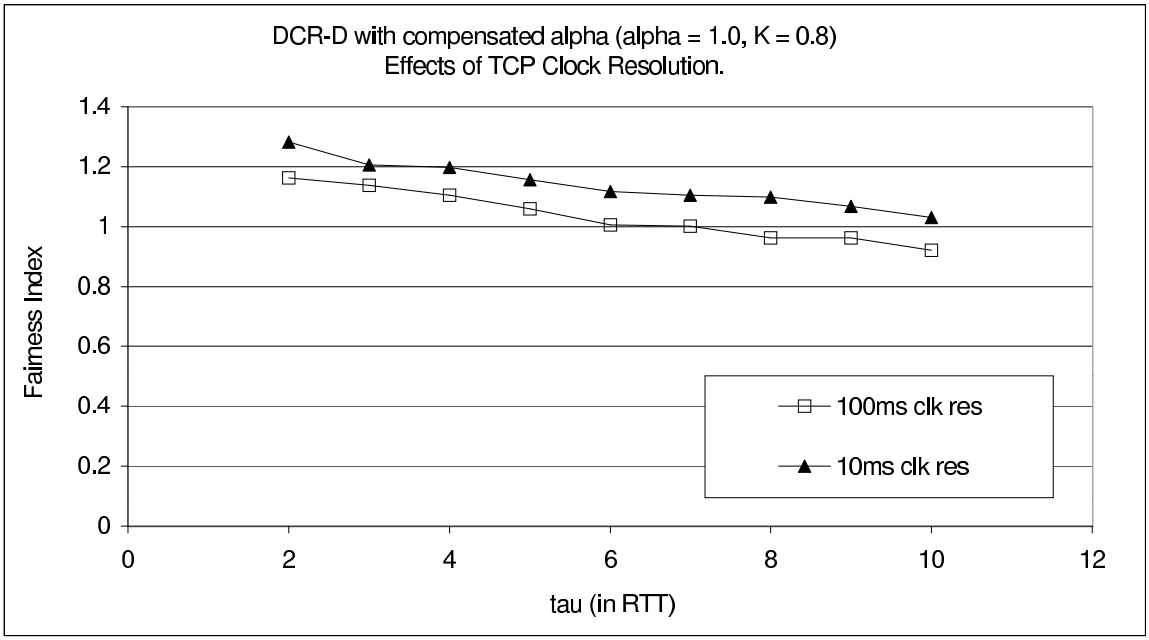


Fig. 20. DCR-D: Effects of Clock Resolution with Compensated Value of α

6. Smoothness of DCR-D protocol

Here we present a comparison between the co-efficient of variance for DCR-D protocol and TCP-Reno. The coefficient of variance was computed as mentioned for the DCR-I protocol.

In the first experiment we took the throughput readings every 0.5 seconds and computed the co-efficient of variance for different values of the droprates. The number of flows in the topology was fixed at 16 and the bandwidth was varied to obtain the different droprates. The buffer size was set to 3*delay*bandwidth of the bottleneck link. The DCR-D parameters were, $K = 0.8$, $\alpha = 0.333$ and $\tau = 5$. An RTT of 100ms was used and the clock resolution was 10ms. Coefficient of variance was computed for each flow. These coefficients of variances were then averaged for each type of flow. Table VIII shows the results. It may be noted that the coefficient of variance of DCR-D is less than that for TCP-Reno for most values of p and is particularly so at low and high values of p.

Table VIII. DCR-D: Coefficient of Variance at Different Droprates

bottleneck link droprate (%)	TCP-reno COV	DCR-D COV
1.7	0.4234	0.3628
2.1	0.4345	0.3898
2.5	0.4620	0.4277
3.1	0.4710	0.4570
4.1	0.4883	0.5020
4.9	0.5283	0.5171
5.8	0.5734	0.5541
6.9	0.6861	0.6359
8.2	0.8060	0.7243
12.0	0.8006	0.7243

In the second experiment we kept the droprate fixed and varied the value of δ.

Again 16 flows were used. The bandwidth of the bottleneck link was fixed at 50Mbps. Rest of the parameters were the same as the previous experiment. A droprate of 0.1% was observed at the bottleneck link. The results as shown in Table IX indicate that DCR-D has lesser variance than TCP-Reno.

Table IX. DCR-D: Coefficient of Variance at Different Values of δ

delta (in seconds)	TCP-reno COV	DCR-D COV
0.1	0.3725	0.2574
0.5	0.3184	0.1786
1	0.2961	0.1719
2	0.2581	0.1654
5	0.1960	0.1518
10	0.1511	0.1329
15	0.1242	0.1218

These results indicate that DCR-D is smoother than TCP-Reno. A direct comparison with the results presented in [6] indicate that DCR-D has a better smoothness than TFRC especially at large droprates. This is a very interesting prospect because DCR-D is much easier to implement than TFRC and can be incorporated into the existing TCP implementations with an addition of another timer to keep track of delayed response.

7. Simulations with only DCR-D flows

In order ot investigate the effects of having only DCR-D flows in a network, we conducted the following set of simulations. The bottleneck link bandwidth was set to 10Mbps and the end-to-end round trip time was set to 100ms. The number of flows in the network was varied to obtain different droprates. Fig. 21 shows the results. In the first graph the normalized throughput of each flow is plotted with the solid line

indicating the average normalized per-flow throughput. In the second graph we show the link utilization at the bottleneck link. It may be noted that for a wide range of droprates, the flows share the bandwidth fairly while maintaining the utilization of the bottleneck link close to 100%.

8. Simulations with different RTTs, but fixed response delays

This experiment was conducted to understand the effects of different flows having different RTTs while their response delays were kept constant. There were a total of 16 flow, 8 of which were TCP-Reno and the other 8 were DCR-D. Out of the 8 flows of each type, 4 had a low end-to-end RTT (25ms in the first experiment and 50ms in the second experiment) and the other 4 had higher RTT (75ms in the first experiment and 100ms in the second experiment). The parameter τ for the DCR-D flows was set such that the total response delay (RTT * τ) was constant for all DCR-D flows. The bottleneck link bandwidth was 10Mbps, utilization was more than 99% and the droprates observed in the first experiment were in the range 3.5 - 5.6% and for the second experiment in the range 1.8 - 2.0 %. Table X shows the results. It may be noticed that the RTT bias of the DCR-D flows is less than the RTT bias of the TCP-Reno flows, which raises the question, of whether DCR-D flows can be designed to get rid of the RTT bias between different flow.

E. Conclusions

The steady state analysis for DCR-D shows that for this protocol to be TCP-friendly, the value of parameter K needs to be set to 1, which violates the basic assumption that $f_2(t)$ is a decreasing function. However a closer look at the condition reveals that K can be set in the range of 0.5 to 0.99 at the cost of introduction of slight

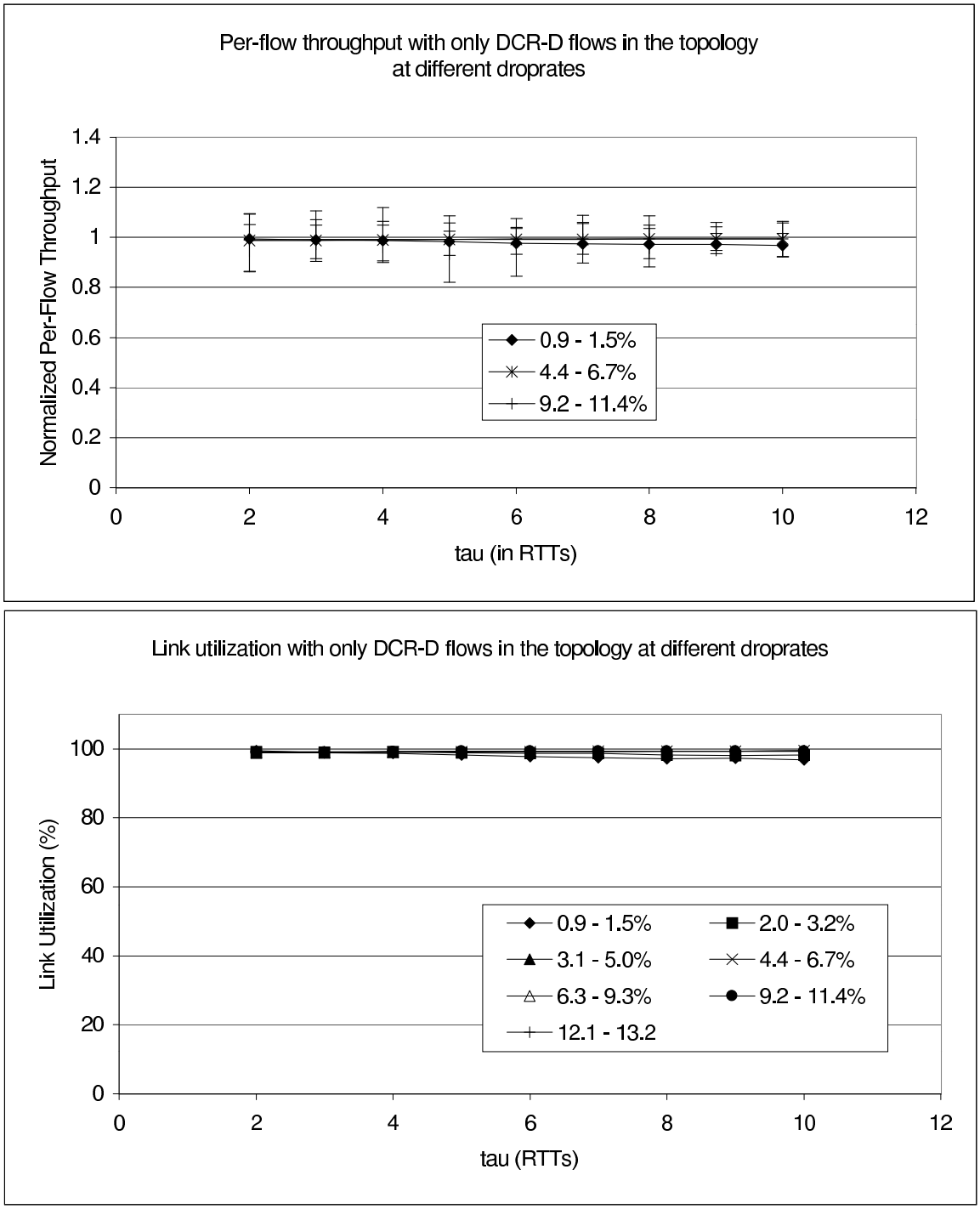


Fig. 21. DCR-D: Simulation Results with Only DCR-D Flows in the Network

Table X. DCR-D: Average Normalized Throughput for Flows with Different RTTs

Response Delay (tau * RTT)	Average Normalised Per-Flow Throughput					
	TCP-reno (RTT = 25ms)	TCP-reno (RTT = 75ms)	$Reno_{25}$ / $Reno_{75}$	DCR-D (RTT = 25ms)	DCR-D (RTT = 75ms)	$DCR\text{-}D_{25}$ / $DCR\text{-}D_{75}$
150ms	1.608	0.546	2.943	1.295	0.515	2.517
225ms	1.678	0.556	3.016	1.219	0.510	2.388
300ms	1.722	0.577	2.987	1.185	0.479	2.473

Response Delay (tau * RTT)	Average Normalised Per-Flow Throughput					
	TCP-reno (RTT = 50ms)	TCP-reno (RTT = 100ms)	$Reno_{50}$ / $Reno_{100}$	DCR-D (RTT = 50ms)	DCR-D (RTT = 100ms)	$DCR\text{-}D_{50}$ / $DCR\text{-}D_{100}$
200ms	1.309	0.732	1.788	1.161	0.771	1.507
300ms	1.376	0.767	1.794	1.078	0.754	1.430
400ms	1.425	0.796	1.791	1.035	0.715	1.447

error. Simulations show that the error is negligible under most circumstances for values of τ less than 7. Also, the clock resolution used for computing the RTT makes a difference. At lower clock resolutions, the DCR-D protocol gets lesser than its fair share of bandwidth. If it is known ahead of time that one of these conditions prevail, then DCR-D performance can be optimized by setting te value of the parameter α higher than indicated by the equation. Preliminary results show that DCR-D has lesser variation in the sending rate compared to TCP-Reno. This added with the choice of providing an early warning regarding an impending reduction in sending rate, makes it an ideal candidate for use with real-time audio video application. However, it can be implemented only in the presence of an active queue management scheme like RED.

CHAPTER V

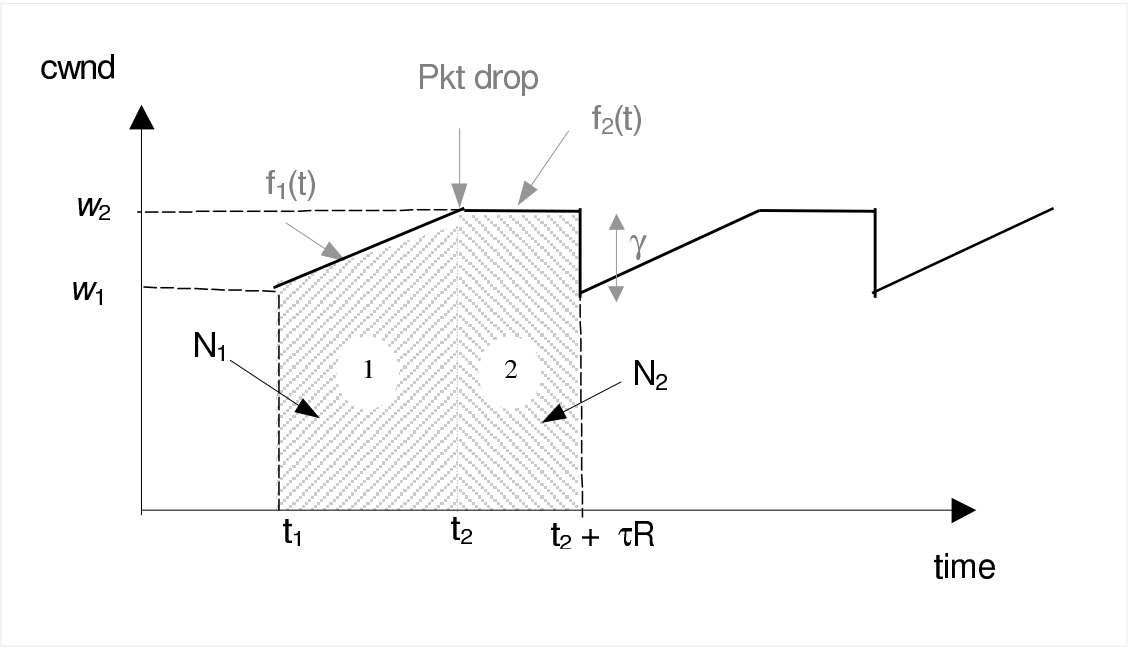
DCR-C: USING A CONSTANT FUNCTION FOR $f_2(t)$

In this algorithm, during the time interval of τ RTTs after a packet drop the congestion window is held constant at the value it had reached just before the packet drop occured. At the end of τ RTTs it is reduced by a factor of γ. The equations are as follows

$$\begin{aligned} f_1(t) \quad &: w_{t+R} \leftarrow w_t + \alpha; \quad \alpha > 0 \\ f_2(t) \quad &: w_{t+R} \leftarrow w_t; \quad t_{drop} < t < t_{drop} + \tau \\ &w_{tdrop+\tau} \leftarrow \gamma * w_{tdrop+\tau-\epsilon}; \quad \gamma < 1 \end{aligned} \tag{5.1}$$

A. Simple Analysis

In this section we present some details of the analysis of this protocol. The analysis was done along the same lines as that of DCR-I and DCR-D.



The throughput of a flow using DCR-C can be given by the equation

$$\begin{aligned} \lambda &= \frac{N_1 + N_2}{(t_2 - t_1) + \tau R} \\ &= \frac{\frac{1}{p}}{(t_2 - t_1) + \tau R} \end{aligned} \tag{5.2}$$

From the equation (3.4) we have

$$\begin{aligned} t_2 - t_1 &= \frac{R}{\alpha}(w_2 - w_1) \\ But, \quad w_1 &= \gamma w_2 \\ t_2 - t_1 &= \frac{R}{\alpha} w_2 (1 - \gamma) \end{aligned} \tag{5.3}$$

By substituting this in equation (5.2) we get

$$\lambda = \frac{1}{\frac{Rp}{\alpha} w_2 (1 - \gamma) + \tau R p} \tag{5.4}$$

Since the functions $f_1(t)$ are the same from DCR-C and DCR-D, the number of packets N_1 sent in the shaded region (1) remains the same as in DCR-D and is given in equation (4.8) to be

$$\begin{aligned} N_1 &= \frac{\alpha}{2R^2}\left(t_2^2 - t_1^2\right) \\ &= \frac{1}{2\alpha} w_2^2 (1 - \gamma^2) \end{aligned} \tag{5.5}$$

From the figure (give fig no.) it is obvious the $N_2 = w_2 \tau$. Since $N_1 + N_2$ gives the total number of packets between two drops, it is equal to 1/p. Thus we have

$$\begin{aligned} N_1 + N_2 &= \frac{1}{p} \\ \frac{1}{2\alpha} w_2^2 \left(1 - \gamma^2\right) + w_2 \tau - \frac{1}{p} &= 0 \end{aligned} \tag{5.6}$$

Solving the above quadratic equation, we obtain the value of w_2 as

$$w_2 = \frac{-\alpha\tau}{(1-\gamma^2)} + \frac{\sqrt{\alpha^2\tau^2 + (1-\gamma^2)\frac{2\alpha}{p}}}{(1-\gamma^2)} \tag{5.7}$$

Substituting this in equation (5.4) we get

$$\lambda = \frac{1}{\frac{-Rp\tau}{(1+\gamma)} + \frac{Rp}{\alpha(1+\gamma)}\sqrt{\alpha^2\tau^2 + (1-\gamma^2)\frac{2\alpha}{p}} + \tau Rp} \tag{5.8}$$

With the above equation we can make two simplifying assumptions. First assumption would be that $\alpha^2\tau^2 \ll (1-\gamma^2)\frac{2\alpha}{p}$. With this assumption, in order to make the DCR-C protocol TCP-friendly, we need to have $\frac{-Rp\tau}{(1+\gamma)} + \tau Rp = 0$. This indicates that the value of the parameter γ should be set to 0, which is very unreasonable. The second assumption that we could make here would be that the term under the square root in the denominator can be simplified using $\sqrt{A^2+B^2} = A + B$. With this assumption, in order that the protocol should be TCP-friendly, we get the condition, $\tau Rp = 0$, which indicates that τ should be 0, which again is unreasonable. Therefore we conclude that for the values of $f_1(t)$ and $f_2(t)$ that we have chosen, DCR-C cannot compete effectively with TCP. In order to confirm that DCR-C indeed cannot survive in the presence of competing TCP flows we conducted two sets of simulations. In the first experiment the DCR-C parameters were fixed at $\alpha = 1$ and $\gamma = 0.5$ and the experiments were repeated for different values of τ and droprates. The Bottleneck link bandwidth was set to 10mbps, RTT to 36ms and the buffers to 3*delay*bandwidth of the bottleneck link. The number of flows was varied to obtain different droprates. Half of the flows were TCP-Reno and the rest were DCR-C. Fig. 22 shows the results.

In the second experiment, we kept the value of τ fixed at 6 and the droprates in the range 3.0-4.0%, and varied the DCR-C parameters. The results are shown in Table XI.

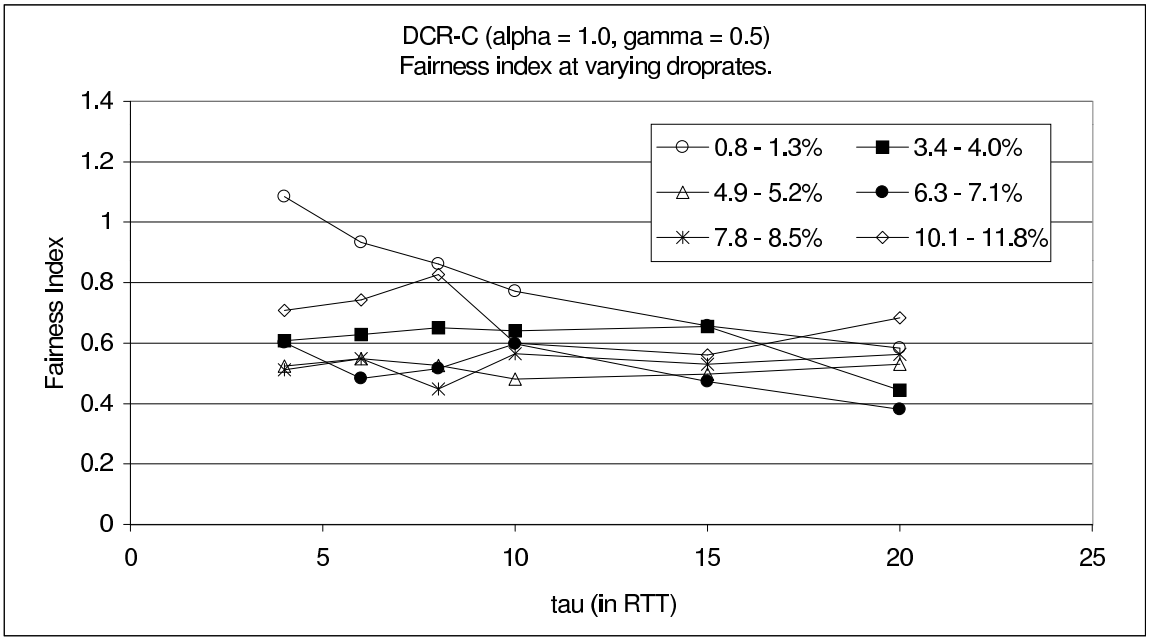


Fig. 22. DCR-C: Fairness Index at Different Droprates

Table XI. DCR-C: Fairness Index at Different Values of DCR-C Parameters

alpha	gamma	Fairness Index
0.1579	0.9	0.7357
0.3333	0.8	0.7886
0.5294	0.7	0.8186
0.7500	0.6	0.6840
1.0000	0.5	0.6796
1.2857	0.4	0.6174
1.6150	0.3	0.3831

B. Conclusions

Through the analysis of DCR-C protocols we have showed that for the functions $f_1(t)$ and $f_2(t)$ that we have taken, DCR-C cannot compete with TCP-Reno for bandwidth. We conducted two sets of simulation that show that this is indeed the case. Further simulations were not conducted with this flavor of Delayed Response Protocols.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this thesis we have laid the ground work for a novel approach for responding to congestion in the network. We have provided the general frame work for a new class of protocols that we call *delayed congestion response protocols*. Our work has been concentrated on three specific cases where the response increases, decreases or remains constant during the period τ when the congestion is delayed. The parameters and functions we chose were heuristics aimed at keeping things simple and easy for analysis and implementation. Substantial amount of work still needs to be done to provide guidelines for choosing these functions and parameters. Also, the general functions that we have provided could be synthesized to find the relationship between the functions $f_1(t)$ and $f_2(t)$ in order to make the response smooth, with minimal variations.

One of the secondary goals when we started out on the thesis was to be able to understand and characterize TCP-friendliness. Our conclusion regarding TCP-friendliness is that steady state analysis of a protocol showing that throughput α $1/\sqrt{p}$ is a necessary but not sufficient condition for the protocol to be TCP-friendly in a practical situation. The first indication towards this is that protocols showing $\lambda \quad \alpha \quad 1/\sqrt{p}$ do not stay TCP-friendly, when passive buffer management schemes are used. This is in agreement with the observation made in [8] that TCP-friendliness does not necessarily mean TCP-compatibility. This is because the AIMD protocols aggressively seeks bandwidth and drastically reduces the sending rate when there is congestion. However, the newer protocols, in the quest of smoothing the variations in the sending rate, increase their sending rate less aggressively and also reduce the rate

less drastically on congestion notification. This results in additional drops and hence and increased average drop probability for competing flows invalidating the basic assumption of the steady state analysis that both the protocols have the same drop probability and rendering the protocol to be no longer TCP-friendly. Even with an active queue management like RED, misconfiguration of the parameters could push it to act like droptail router, in which case the protocols cease to be TCP-friendly.

One of the other puzzling observations is that with the DCR-D protocol, when a mixed load was used, the protocol seems to fare well as long as there were *large* numbers of flows using DCR-D protocol. However, when there were fewer flows using DCR-D protocol, it managed to claim much more than its fair share of bandwidth. This indicates that TCP-friendliness is a far more complex matter than expected. Last but not the least, if the new protocols use timers for scheduling their increase or decrease, then the TCP-friendliness will be affected by the granularity of time calculations.

There has been a general conception in literature that if the steady state analysis shows that the protocol has the property $\lambda \quad \alpha \quad 1/\sqrt{p}$ it can be considered TCP-friendly. However our observation is that this is not true. More work needs to be done in the future to decide a more quantitative measure for TCP-friendliness.

REFERENCES

[1] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of ACM SIGCOMM '88,* Stanford, CA, August 1988, pp. 314–329.

[2] D. Tan and A. Zakhor, "Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol," *IEEE Transactions on Multimedia,* vol. 1, no. 2, pp. 172–186, May 1999.

[3] S. Floyd, "Congestion Control Principles," Internet Engineering Task Force, RFC 2914, Best Current Practice, September 2000.

[4] S. Floyd and K. Fall, "Router Mechanisms to Support End-to-End Congestion Control," Technical Report, Lawrence Berkeley Laboratory, Berkeley, CA, February 1997.

[5] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation," in *Proceedings of ACM SIGCOMM '98,* Vancouver, Canada, September 1998, pp. 303–314.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation Based Congestion Control for Unicast Applications," in *Proceedings of ACM SIGCOMM 2000,* Stockholm, Sweden, August 2000, pp. 43–56.

[7] D. Sisalem and A. Wolisz, "LDA+ TCP-Friendly Adaptation: A Measurement and Comparison Study," in *10th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'2000),* Chapel Hill, NC, June 2000.

[8] D. Bansal and H. Balakrishnan, "TCP-Friendly Congestion Control for Real-time Streaming Applications," Technical Report MIT-LCS-TR-806, MIT Laboratory for Computer Science, Cambridge, MA, May 2000.

[9] J. Mahdavi, and S. Floyd, "The TCP-Friendly Website," http://www.psc.edu/networking/tcp_friendly.html, Pittsburgh Supercomputing Center, Carnegie Mellon University, Pittsburgh, PA, 1998.

[10] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking,* vol. 7, no. 4, pp. 458–472, August 1999.

[11] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking,* vol. 1, no. 4, pp. 397–413, August 1993.

[12] T. V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Transactions on Networking,* vol. 5, no. 3, pp. 336–350, 1997.

[13] J. Padhye, J. Kurose, D. Towsley and R. Koodli, "A Model Based TCP-Friendly Rate Control Protocol," in *9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '99),* Basking Ridge, NJ, July 1999, pp. 137–151.

[14] "ns-2 Network Simulator," Dept. of EECS, University of California at Berkeley, CA. Available at http://www.isi.edu/nsnam/

[15] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan,

S. Shenker, and J. Wroclawski, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, Informational, April 1998.

[16] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proceedings of IEEE INFOCOM,* New York, NY, March 1999, pp. 1337–1345.

[17] B. Smith, "Cyclic-UDP: A Priority-Driven Best-Effort Protocol," Technical Report, Cornell University, Ithaca, NY, 1996.

[18] D. Dwyer, S. Ha, J-R. Li and V. Bhargavan, "An Adaptive Transport Protocol for Multimedia Communication," in *IEEE International Conference on Multimedia Computing Systems*, Austin, TX, June 1998, pp. 23–32.

[19] J. Salehi, Z. L. Zhang, J. Kurose and D. Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements Through Optimal Smoothing," in *Proceedings of ACM SIGMETRICS*, Philadelphia, PA, May 1996, pp. 222–231.

VITA

Sumitha Bhandarkar was born on February 4, 1976 in Manipal, India. She received her Bachelor of Engineering degree in electronics and communication engineering from Mysore University, India in August 1997. She worked for two years at Tata Infotech Ltd., Bangalore, India. In August 1999, she started her Master's program in Computer Engineering at Texas A&M University. Her research at Texas A&M University has been focused on networking for multimedia applications. Her address is Department of Electrical Engineering, Texas A&M University, College Station, TX 77843-3128.

The typist for this thesis was Sumitha Bhandarkar.