A Game-Theoretic Approach Towards Congestion Control in Communication Networks

Rahul Garg*
garg@ieee.org

Abhinav Kamra†
abhinav@iitd.ernet.in

Varun Khurana‡
varunk@netapp.com

ABSTRACT

Most of the end-to-end congestion control schemes are "voluntary" in nature and critically depend on end-user cooperation. We show that in the presence of selfish users, all such schemes will inevitably lead to a congestion collapse. Router and switch mechanisms such as service disciplines and buffer management policies determine the sharing of resources during congestion. We show, using a game-theoretic approach, that all currently proposed mechanisms, either encourage the behaviour that leads to congestion or are oblivious to it.

We propose a class of service disciplines called the Diminishing Weight Schedulers (DWS) that punish misbehaving users and reward congestion avoiding well behaved users. We also propose a sample service discipline called the Rate Inverse Scheduling (RIS) from the class of DWS schedulers. With DWS schedulers deployed in the network, max-min fair rates constitute a unique Nash and Stackelberg Equilibrium. We show that RIS solves the problems of excessive congestion due to unresponsive flows, aggressive versions of TCP, multiple parallel connections and is also fair to TCP.
Keywords: Game Theory, Nash Equilibrium, Stackelberg Equilibrium, Generalized Processor Sharing, GPS, Scheduling, Congestion Control, TCP, Fairness, RIS, DWS.

1. INTRODUCTION

Most of the end to end congestion control schemes [23, 17, 32, 26, 16] are voluntary in nature and critically depend on end-user cooperation. The TCP congestion control algorithms [23, 5, 20, 21, 19, 9] voluntarily reduce the sending rate upon receiving a congestion signal such as ECN [25], packet loss [14, 10, 18] or source quench [24]. Such congestion control schemes are successful because all the end-users cooperate and volunteer to reduce their sending rates using similar algorithms, upon detection of congestion.

As the Internet grows from a small experimental network to a large commercial network, the assumptions about cooperative end-user behaviour may not remain valid. Factors such as diversity, commercialization and growth may lead to non-cooperative and competitive behaviours [12] that aim to derive better individual utility out of the shared Internet resources.

If an end-user does not reduce its sending rate upon congestion detection, it can get a better share of the network bandwidth. The flows of such users are called unresponsive flows [8, 7]. Even responsive flows that react to congestion signal can get unfair share of network bandwidth by being more conservative in reducing their rates and more aggressive in increasing their rates. Such flows are termed as TCP-incompatible flows [8, 7]. Even TCP-compatible flows such as different variants of TCP give different performance [21] under different conditions. Such behaviours though currently not prevalent, are present in the Internet, and pose a serious threat to Internet stability [12, 8]. If widespread, such behaviours may lead to a congestion collapse of the Internet (see Section 2). Therefore it is important to have an approach towards congestion control that is not dependent on cooperative end users voluntarily following an end-user behaviour from a class of predefined behaviours.

In this paper we propose a game-theoretic approach towards congestion control. The crux of the approach is to deploy schedulers and/or buffer management policies at intermediate switches and routers that punish misbehaving flows by cutting down their rates thereby encouraging well behaved flows. There have been discussions [29] on punishing misbehaving users but they do not talk about such punishments in a game-theoretic framework. We propose a class of scheduling algorithms called the Diminishing Weight Scheduling (DWS) that punish misbehaving flows in such a way that the resulting game-theoretic equilibrium (Nash Equilibrium) results in fair resource allocations. We show that with DWS scheduling deployed in the network, max-min fair rates [4] constitute a Nash as well as a Stackelberg Equilibrium. Thus, with DWS scheduling deployed, the "best selfish behaviour" for each user is to estimate its fair rate and send traffic at that rate.

Our game-theoretic approach is very similar to that proposed by Shenker [27] to analyze switch service disciplines. However, Shenker uses a discrete queueing theoretic model of input traffic which does not accurately model the traffic in today's data networks. Moreover, Shenker's analysis is restricted to a single switch/router and does not extend to

*IBM India Research Lab, Hauz Khas, New Delhi - 110016, INDIA.
†Indian Institute of Technology, Delhi, Hauz Khas, New Delhi - 110016, INDIA.
‡Network Appliance, CA, USA.

an arbitrary network in a straight forward manner. We use a continuous fluid-flow based input traffic model which is more realistic and amenable to analysis in an arbitrary network. This makes our approach more practical and applicable to networks such as the Internet.

The steepness of the diminishing weight function determines the amount of penalty imposed on a misbehaving flow. Steeper weight functions impose stricter penalties to misbehaving flows. As the weight function becomes flat, DWS approaches WFQ scheduling which imposes no penalty on a misbehaving flow. We also present a sample service discipline called Rate Inverse Scheduling (RIS) where the diminishing weight function is the inverse of the input rate. By using different weight functions in DWS, the switch designers and ISPs can choose from a variety of reward-penalty profiles to meet their policy requirements.

With DWS deployed, we show that it is in the interest of each individual end-user to follow a TCP-style increase/decrease algorithm. Using simulations we show that end-users using different versions of TCP are actually able to converge to their fair rates, even in the presence of misbehaving users.

In Section 2 we present our game-theoretic formulation and show that in the presence of selfish users, the current resource management policies will lead to a congestion collapse. In Section 3 we present the DWS scheduling algorithm and discuss its properties in Section 4. We present some preliminary simulation results in Section 5. We conclude in Section 6. The proofs are provided in the Appendix.

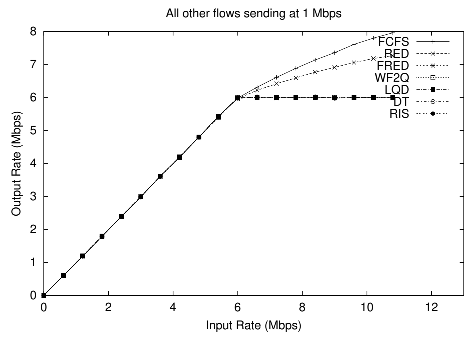

Figure 1: Uncongested link

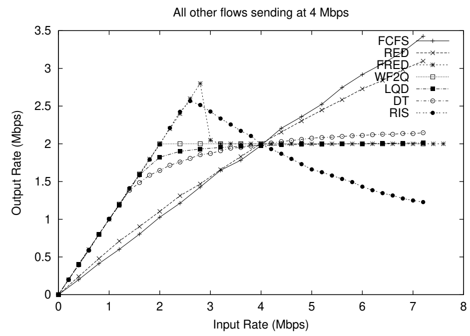

Figure 2: Congested link

2. A GAME-THEORETIC MODEL OF A NETWORK

Consider a link of capacity C shared by N users. There is a sufficiently large shared buffer, a buffer management policy, and a service discipline to partition the link capacity among the users. Assume that user i sends a constant rate traffic flow at a rate r_i (the input rate). Some of this traffic may be dropped due to buffer overflows. Assume that, in steady state the traffic of user i is delivered at the destination with an average output rate γ_i, ($\gamma_i \leq r_i$). The output rate is a function of sending rate of all the N users, the switch service discipline S, and the buffer management policy B. Mathematically, this is written as $\underline{\gamma} = \underline{\gamma}^{SB}(\underline{r})$, where $\underline{r} = (r_1, r_2, \ldots, r_N)$ denotes the N-dimensional vector of input rates and $\underline{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_N)$ denotes the N-dimensional vector of output rates and $\underline{\gamma}^{SB}()$ is the function (called the resource management function) dependent on scheduling discipline S and buffer management policy B mapping the vector of input rates to the vector of output rates.

Consider a network comprising multiple nodes and links. Assume that traffic of user i traverses links $l_1, l_2, \ldots, l_k$. Let the resource management function at link l_j be $\underline{\gamma}^j()$. Let $\underline{r}^j$ denote the vector of input rates of all users at link l_j. The input rate of user i at link l_1 is $r_i^1 = r_i$. Since we assume that all users send traffic at a constant rate, the output rate of user i at link l_1 is also a constant given by $\gamma_i^1 = \underline{\gamma}_i^1(\underline{r}_1)$. Therefore, the input rate of user i at link l_2 will also be constant given by $r_i^2 = \gamma_i^1$. Similarly we have:

$$r_i^j = \gamma_i^{j-1} = \underline{\gamma}_i^{j-1}(\underline{r}^{j-1}). \tag{1}$$

The final output rate of the user will be given by:

$$\gamma_i = \gamma_i^k = \underline{\gamma}_i^{k-1}(\underline{r}^{k-1})$$

.

In general, a user's utility (or satisfaction) depends on its output rate, loss rate and end-to-end delay. However, for a majority of applications the output rate is the most important factor determining the user's utility. For instance, "fire-hose applications" described in [29] are completely loss tolerant. For streaming media applications, loss tolerance can be obtained using forward error correction techniques [2]. For bulk transfer applications, loss tolerance can be achieved using selective retransmissions [4]. Therefore, for simplicity, we assume that a user's utility is an increasing function of its output rate only. The class of such utility functions U^γ, is formally defined as[1]:

1. $U \in U^\gamma$ maps a user's output rate γ to a real-valued non-negative utility,
2. $U(\gamma') > U(\gamma)$ iff $\gamma' > \gamma$.

If user i was to act in a selfish manner, it would choose a sending rate r_i which would maximize its utility (and hence the output rate), irrespective of the amount of inconvenience caused (loss of utility) to other users. Consider what will happen in such a scenario with different packet service disciplines and buffer management policies.

Consider a link of capacity $C = 10Mbps$ shared by five users sending traffic at a constant rate. Figures 1 and 2

[1]Later, in Section 4 we also consider the class of utility function $U^{\gamma l}$ where a user's utility is also dependent on its loss rate.

Legend	Scheduling discipline	Buffer management policy
FCFS	First come first serve	Shared buffers, drop tail
WF2Q	Worst case fair weighted fair queueing	Per flow buffers, drop tail [3]
LQD	First come first serve	Longest queue tail drop [30]
DT	First come first serve	Dynamic threshold [6]
RED	First come first serve	Random early drop [10]
FRED	First come first serve	Flow RED [18]
RIS	Rate inverse scheduling	Per flow buffers, drop tail

Table 1: Notations for schedulers and buffer management policies

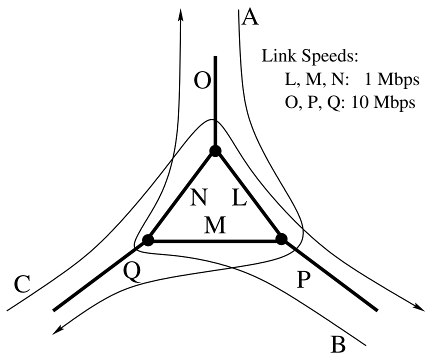

Figure 3: Congestion collapse in a sample network

show the variation in a user's output rate as a function of its input rate for different scheduling disciplines and buffer management policies. Refer to Table 1 for notations.

For FCFS, the output rate of a user (and hence the user's utility) always increases as its input rate increases. However, the slope of the graph depends on the input rate of other users. In such a case, there is an incentive for each user to increase its sending (input) rate, irrespective of what other users are doing. If each user was to act selfishly, to maximize its own utility, the link will end up becoming heavily congested with each user sending traffic at its maximum possible rate and receiving only a tiny fraction of the traffic sent. This is characterized by the concept of Nash Equilibrium [11].

DEFINITION 1 (NASH EQUILIBRIUM). *Let $u_i(\sigma, \underline{\sigma})$ represent the utility of player i when the player adopts the strategy σ and all the other players adopt the strategy $\underline{\sigma}$. A strategy profile $\underline{\sigma}^*$ is a Nash Equilibrium if, for every player i,*

$$u_i(\sigma_i^*, \underline{\sigma}_{-i}^*) \geq u_i(\sigma_i, \underline{\sigma}_{-i}^*),$$

for all $\sigma_i \in S_i$, where S_i is the set of strategies that user i can adopt.

In other words, a Nash Equilibrium is a strategy profile where no user has an incentive to deviate unilaterally from its strategy. If all the users are selfish and non-cooperating they would eventually adopt strategies that constitute a Nash Equilibrium.

Here, the vector of input rates $\underline{r}$ constitutes a strategy profile, and each user's utility $U(\gamma)$ is a monotonically increasing function of its output rate γ. For FCFS, RED, and DT resource management policies, the only Nash Equilibrium is when the input rates approach infinity.

Therefore, it is appropriate to say that FCFS encourages behaviour that leads to congestion. In a network comprising multiple nodes and links, selfish user behaviour will lead to worse disasters [7, 8] (similar to the congestion collapse) where input rate of each user will approach their maximum possible and output rates will approach zero. To see this, consider the network and flows shown in Figure 3. Assume that FCFS policy is deployed at every link. Every user will send traffic at the rate of the access link, 10 Mbps, and will get a net output rate of less than 100Kbps at the final destination.

Now consider WF2Q. The output rate of a user remains equal to its input rate as long as it is less than or equal to its fair rate. When, the input rate becomes larger than the fair rate, the output rate remains constant at fair rate (C/N). Above is true irrespective of the input rates of other users. In such a scenario, a selfish user will increase its input rate till the fair rate. However, a user has no incentive to increase its input rate beyond the fair rate, nor does it have any incentive to keep its input rate down to the fair rate. Therefore, in a network comprising multiple nodes and links, when a selfish user neither knows its fair rate, nor the resource management policies employed (FCFS or WFQ), it may simply find it convenient to keep on increasing its input rate much beyond the fair rate. For WF2Q, LQD, and FRED, it seems that any vector of input rates where each user's input rate is more than C/N constitutes a Nash Equilibrium. We say that such policies are oblivious to congestion causing behaviour.

Observe from Figures 1 and 2 that all the resource management policies (except RIS which will be described in following sections) either encourage behaviour leading to congestion or are oblivious to it.

For end-to-end congestion control schemes to be effective in the presence of selfish users (and in the absence of other incentives such as usage based charging, congestion pricing etc.), a resource management mechanism in the interior of the network (i.e., the traffic police) is needed that punishes misbehaving users and rewards well behaved users, while what is present in today's networks is just the opposite. In the following section we describe a class of service disciplines that achieve the purpose of rewarding the well behaved users and punishing the misbehaving users.

3. DIMINISHING WEIGHT SCHEDULERS

The class of Diminishing Weight Schedulers (DWS) is defined for the idealized fluid-flow traffic model. It is derived from the Generalized Processor Sharing (GPS) scheduling algorithm [22]. Consider a link of capacity C shared by N users sending traffic as N distinct flows. Let $A_i(t)$ be the amount of traffic of flow i entering the scheduler buffer in the interval $(0, t]$ and $S_i(t)$ be the amount of traffic of flow i served by the scheduler in the interval $(0, t]$. Define

$A_i(0) = 0$ and $S_i(0) = 0$ for all i. Define the backlog of flow i at time $t > 0$ as $B_i(t) = A_i(t) - S_i(t)$. Define the total system backlog at time t as $B(t) = \sum_{i=1}^{N} B_i(t)$. Define the input rate of a flow at the link at time t as $r_i(t) = dA_i(t)/dt$ and define the output rate of flow i at the link at time t as $\gamma_i(t) = dS_i(t)/dt$.

A GPS scheduler [22] on a link may be defined as the unique scheduler satisfying the following properties:

Flow Conservation:

$$B_i(t) \geq 0, \forall i, t \geq 0. \tag{2}$$

Work Conservation:

$$\text{If } B(t) > 0, \text{ then } \sum_{i=1}^{N} \gamma_i(t) = C. \tag{3}$$

GPS Fairness:

$$B_i(t) > 0 \Rightarrow \forall j : \frac{\gamma_i(t)}{\phi_i} \geq \frac{\gamma_j(t)}{\phi_j}, \tag{4}$$

where ϕ_i is the GPS weight assigned to flow i.

The flow conservation property implies that for a flow, the traffic served cannot be more than the traffic arrived. The work conservation property implies that if there is a non-zero backlog in the system, then the link is not kept idle. The fairness property implies that the output (service) rates of all the backlogged flows will be proportional to their respective GPS weights, while the output rates of non-backlogged flows will be smaller.

GPS assigns constant weights to all the flows. DWS differ from GPS in this regard. In DWS, each bit gets a GPS weight that is a decreasing function of that bit's arrival (input) rate. If the bit at the head of the queue of flow i at time t arrived at time $\tau_i(t)$, then in DWS, $\phi_i(t) = \theta_i W(r_i(\tau_i(t)))$, where $W(r)$ is the diminishing weight function, which is a continuous and strictly decreasing function of r. The class of DWS schedulers is formally defined as the schedulers satisfying the following properties:

Flow Conservation:

$$B_i(t) \geq 0, \forall i, t \geq 0. \tag{5}$$

Work Conservation:

$$\text{If } B(t) > 0, \text{ then } \sum_{i=1}^{N} \gamma_i(t) = C. \tag{6}$$

DWS Fairness:

$$\begin{array}{l} B_i(t) > 0 \Rightarrow \\ \quad \forall j : \frac{\gamma_i(t)}{\theta_i . W(r_i(\tau_i(t)))} \geq \frac{\gamma_j(t)}{\theta_j . W(r_j(\tau_j(t)))}, \end{array} \tag{7}$$

where θ_i is the DWS weight for flow i. Thus, DWS rewards flows with small rates by assigning them large GPS weights and punishing flows with large rates by assigning them small GPS weights. The amount of punishment depends upon the steepness of the diminishing weight function. If it is a flat function such as the $1/log(r)$ function, then DWS resembles the GPS scheduling. If the diminishing weight function is steep then strict penalties are enforced to misbehaving users. The DWS weights θ may be set in accordance with the pricing, resource sharing or other administrative policies, in the same way the as GPS weights ϕ are set when GPS based schedulers are deployed.

The Rate Inverse Scheduler (RIS) is a special case of the DWS scheduler where the diminishing weight function is the inverse function ($W(r) = 1/r$). Thus the DWS fairness condition for RIS reduces to the following:

RIS Fairness:

$$\begin{array}{l} B_i(t) > 0 \Rightarrow \\ \quad \forall j : \frac{1}{\theta_i} \gamma_i(t) r_i(\tau_i(t)) \geq \frac{1}{\theta_j} \gamma_j(t) r_j(\tau_j(t)). \end{array}$$

Assume, for simplicity that all DWS weights are set to 1 and all users send traffic at a constant rate. Thus, all output rates will also be constant. From the flow conservation property it follows that $\gamma_i \leq r_i$. The DWS fairness condition can be simplified as follows:

$$B_i(t) > 0 \Rightarrow \forall j : \gamma_i / W(r_i) \geq \gamma_j / W(r_j) \tag{8}$$

It follows that if two flows i and j are backlogged, then

$$\gamma_i / W(r_i) = \gamma_j / W(r_j) = k \text{ (constant)} \tag{9}$$

We now prove some important properties of DWS scheduling. Define the congestion characteristic function $G(x, \underline{r})$ as:

$$G(x, \underline{r}) = \sum_{i=1}^{N} \min\left(x.W(r_i), r_i\right). \tag{10}$$

Define the rate constant κ for a link with a vector of input rates $\underline{r}$ as :

$$\kappa = \begin{cases} \frac{C}{W(C)} & \text{if } \sum_{i=1}^{N} r_i \leq C \\ x : G(x, \underline{r}) = C & \text{if } \sum_{i=1}^{N} r_i > C \end{cases} \tag{11}$$

THEOREM 3.1 (RATE CONSTANT). *Rate constant κ as defined in Eq 11 is unique and the output rate of flow i is uniquely given by $\gamma_i = \min(\kappa.W(r_i), r_i)$.*

The proof is provided in the Appendix. Hence, given the input rates of flows, using the rate constant κ it is possible to uniquely determine the output rate of any flow. We define the fair rate f as follows:

$$f = \{x : x = \kappa.W(x)\} \tag{12}$$

LEMMA 3.1. *The fair rate f as defined in Eq.12 is unique.*

The proof is provided in the Appendix. The output rate γ_i can be represented in terms of the fair rate as follows:

$$\gamma_i = \min(W(r_i)\frac{f}{W(f)}, r_i) \tag{13}$$

We now show that if the input rate of a flow is less than or equal to the fair rate, then the flow will get all its bits transmitted without loss, otherwise it will suffer a loss according to the diminishing weight function $W()$.

LEMMA 3.2. *If $r_i \leq f$ then $\gamma_i = r_i$ else $\gamma_i = \kappa.W(r_i) < f$.*

PROOF. **Case 1** ($r_i \leq f$): Note that $r_i \leq f \Rightarrow W(r_i) \geq W(f)$, since $W()$ is a strictly decreasing function. Hence we get $r_i/W(r_i) \leq f/W(f)$. Using Eq.12 we get $r_i \leq \kappa.W(r_i)$. Therefore, $\gamma_i = \min(\kappa.W(r_i), r_i) = r_i \leq f$.

Case 2 ($r_i > f$): In this case we get $r_i/W(r_i) > f/W(f)$, since $W()$ is a strictly decreasing function. Use Eq.12 to get $r_i > \kappa.W(r_i)$. Therefore, $\gamma_i = \min(\kappa.W(r_i), r_i) = \kappa.W(r_i)$. Since $\kappa = f/W(f)$ and $W(f) < W(r_i)$, we also have $\gamma_i < f$. $\square$

The above behaviour is also evident from Figures 2 and 7. We say that a flow i contributes to a link's congestion iff $r_i > f$. With DWS scheduling, the output rate of a flow remains equal to its input rate as long as the flow does not contribute to congestion. However, as soon as the flow contributes to congestion its output rate begins to decline according to the diminishing weight function $W()$. In DWS, different weight functions can be chosen to meet specific policy requirements. Observe from Figure 7 that the weight function $W(r) = 1/\log(r)$ imposes a very small penalty on misbehaving flows and is very similar to WFQ whereas the weight function $W(r) = 1/r^4$ imposes very strict penalties.

The following result follows from Lemma 3.2

COROLLARY 3.1. *The output rate for any flow i is less than equal to the fair rate ($\gamma_i \leq f$).*

We now establish the relationship between the fair rate f, the link capacity C and the number of users N. It also suggests that if all the users are equal (with equal DWS weights), then the fair rate is indeed fair.

LEMMA 3.3. *Fair rate f is greater than or equal to C/N.*

The proof is provided in the Appendix. From Lemmas 3.2 and 3.3, observe that if a user i's input rate is C/N, its output rate will also be equal to C/N (since $r_i = C/N \leq f$), and hence DWS results in fair allocation of resources.

LEMMA 3.4. *If a flow i is experiencing losses ($\gamma_i < r_i$), then decreasing the flow's input rate by a sufficiently small amount will either increase its output rate, or leave it unchanged (i.e. $\exists \delta > 0$ s.t. if $r'_i = r_i - \delta$ then $\gamma'_i \geq \gamma_i$.*

The proof is provided in the Appendix.

LEMMA 3.5. *If a flow i is not experiencing losses, then increasing the flow's input rate by a sufficiently small amount will either increase its output rate or leave it unchanged (i.e. $\exists \delta > 0$ s.t. if $r'_i = r_i + \delta$ then $\gamma'_i \geq \gamma_i$).*

The proof is provided in the Appendix.

The above two lemmas establish that a flow experiencing losses ($\gamma_i < r_i$) may have an incentive to reduce its input rate, whereas a flow experiencing no losses may have incentive to increase its input rate. This is very similar to the behaviour of TCP's increase/decrease algorithms which increase input rate when there are no losses and decrease the input rate as soon as losses are observed. Later in Section 5 using simulations we show that different versions of TCP actually do converge close to their fair rate when DWS schedulers are deployed.

3.1 Packetized Diminishing Weight Schedulers (PDWS)

In a network, traffic does not flow as a fluid. Instead it flows as packets containing chunks of data that arrive at discrete time boundaries. Therefore, a scheduler is needed that works with discrete packets. Packetized DWS is derived from the DWS scheduler in the same way as packetized GPS is derived from the GPS scheduler. Therefore the implementation details of PDWS are very similar to those of PGPS except for minor changes in equations computing the timestamps. It should be straightforward to adapt PDWS to the simplifications of PGPS like Virtual Clock [34], Self Clocked Fair Queueing [13], WF2Q+ [33], Frame based Fair Queueing (FFQ) [28], etc.

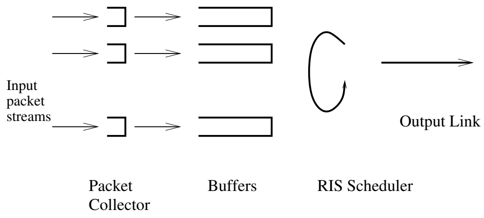

Figure 4: Hypothetical model for DWS with discrete packet boundaries

Denote the arrival time of the k^{th} packet of flow i as a_i^k and the length of the k^{th} packet of flow i as L_i^k. We model the k^{th} arrival of flow i as if it were fluid flowing at a rate $r_i^k = L_i^k/(a_i^k - a_i^{k-1})$ in the interval $(a_i^{k-1}, a_i^k]$. The rate of arrival of all the bits of the packet is given by r_i^k and therefore this packet gets a GPS weight given by $\phi_i^k = \theta_i W(r_i^k)$. RIS being a special case of DWS has $\phi_i^k = \theta_i/r_i^k$ (since $W(r) = 1/r$ for RIS). A packet becomes eligible for service by the scheduler only after its last bit has arrived, i.e., at time a_i^k. We assume that there is a hypothetical packet collector before the DWS scheduler which collects all the bits of a packet and gives them to DWS only when they become eligible (see Figure 4).

Now, we define the finish time of a packet as the time when the last bit of the packet gets serviced in a hypothetical DWS scheduler with a packet collector as shown in Figure 4. PDWS is defined as the scheduler that schedules packets in increasing order of their finish times.

Along the lines of PGPS implementation [31], PDWS is based on the concept of system virtual time and virtual finish time of packets. The scheduler maintains a virtual time function $v(t)$. Upon a packet arrival, each packet is tagged with a virtual finish time as follows:

$$F_i^k = \max\left(F_i^{k-1}, v(a_i^k)\right) + \frac{L_i^k}{\phi_i^k} \quad (14)$$

$$= \max\left(F_i^{k-1}, v(a_i^k)\right) + \frac{L_i^k}{\theta_i W(r_i^k)} \quad (15)$$

The packets are serviced in increasing order of their virtual finish times. To compute the virtual time at any instant, an emulation of hypothetical DWS system of Figure 4 is maintained which is similar to most PGPS implementations. When a packet k of flow i arrives, it is tagged with a GPS weight of ϕ_i^k and is also given to the DWS emulation. The emulation computes the virtual finish time of the packet by Eq 15. The rate of change of virtual time with real time is given by $d(v(t))/dt = C/(\sum_{i=1}^{N} \phi_i)$, where ϕ_i is the GPS weight corresponding to the packet of flow i which is currently in service in the DWS emulation.

In real practice traffic arrivals are bursty. Therefore, it is better to use a smoothed arrival rate, instead of instantaneous arrival rates for GPS weight computation. The scheduler PDWS with α smoothing sets $\phi_i^k = \theta_i W(\hat{r}_i^k)$, where

$\hat{r}_i^k = \alpha \hat{r}_i^{k-1} + (1-\alpha) r_i^k$. The value of α is taken such that the half life of smoothing is of the order of one round trip time (R) when packet size of L_{max} is used to send traffic. This gives:

$$\alpha = 2^{-L_{max}/(fR)} \tag{16}$$

where f is the fair rate of the flow.

4. PROPERTIES OF DWS

We now describe some desirable game-theoretic properties of DWS Schedulers. In this section, for all results number of users, $N \geq 2$ unless otherwise specified.

4.1 Single Link

With DWS scheduling, the best strategy for each individual user is to send traffic at its fair rate. This is formally illustrated in the following theorem.

THEOREM 4.1. *Consider a link of capacity C, shared by N users, using DWS scheduling with unit DWS weights. Then $\forall_i r_i = C/N$ is the unique Nash Equilibrium for the system.*

The proof is provided in the Appendix. The naive selfish users will converge to a Nash Equilibrium. However, in case a user (say a leader) had information about other users' utility functions or behaviours, scheduling and/or queue management policies at the gateways, it could choose a value for its input rate and the other users would equilibrate to a Nash Equilibrium in the $N-1$ user subsystem. The leading user can then choose its input rate based on which of these $N-1$ subsystem Nash Equilibria maximizes the leading user's utility. This is formally called a Stackelberg Equilibrium.

DEFINITION 2 (STACKELBERG EQUILIBRIUM). *A strategy profile $\underline{\sigma}^*$ is a Stackelberg Equilibrium with user 1 leading if:*

1. *it is a Nash Equilibrium for users $2 \dots N$, i.e.* $\forall_{i \in [2,\dots,N]}, \forall \sigma_i \in S_i :$ $u_i(\sigma_1^*, \sigma_i^*, \underline{\sigma}^*_{-1,-i}) \geq u_i(\sigma_1^*, \sigma_i, \underline{\sigma}^*_{-1,-i})$,
2. *the leader's utility is maximized, i.e.* $\forall \sigma_1 \in S_1, u_1(\sigma_1^*, \underline{\sigma}^*_{-1}) \geq u_1(\sigma_1, \underline{\hat{\sigma}}_{-1})$, *where* $\underline{\hat{\sigma}}_{-1}$ *is a Nash Equilibrium for users* $[2 \dots N]$ *when user* 1 *adopts the strategy* σ_1,

where S_i is the set of strategies that user i can follow.

The leader's utility at a Stackelberg Equilibrium is never less than that in any other Nash Equilibrium. So a user with more information may try to drive the system towards one of its Stackelberg Equilibria. This can be avoided if Nash and Stackelberg Equilibria coincide. We now show this to be true for a single link with DWS scheduling.

THEOREM 4.2. *Consider a link of capacity C, shared by N users, using DWS scheduling with unit DWS weights. Then $\forall_i r_i = C/N$ is the unique Stackelberg Equilibrium for the system.*

The proof is provided in the Appendix. Since the unique Nash and Stackelberg Equilibria coincide, a user will benefit most by sending at its fair rate. Any user sending at a rate higher than its fair rate will be penalized, and other users can then receive a better output rate. This is characterized by the concept of Nash rate.

DEFINITION 3. *Given a user with input rate r, define Nash rate for the remaining users as:*

$$\eta(r) = \begin{cases} (C-r)/(N-1) & \text{if } r \leq C/N \\ x \geq 0 : xW(r)/W(x) + \\ \quad (N-1)x - C = 0 & \text{otherwise.} \end{cases} \tag{17}$$

We now discuss some properties of the Nash rate.

LEMMA 4.1. *The Nash rate ($\eta(r)$) is a strictly increasing function of r in the range $(C/N, \infty)$.*

PROOF. For $r \in (C/N, \infty)$, we rewrite definition 17 in the form

$$\eta(r) = x : W(r) = W(x)(\frac{C}{x} - (N-1))$$

Since $W()$ is strictly decreasing, r increases as x increases and vice-versa. Also note that the value of x satisfying the above equation for a given value of r is unique. ☐

LEMMA 4.2. *The Nash rate is greater than or equal to C/N i.e. $\eta(r) \geq C/N$.*

PROOF. For $r \leq C/N$, we see from the definition of Nash rate (Eq. 17) that $\eta(r) \geq C/N$.

For $r > C/N$, note from Lemma 4.1, that $\eta(r)$ is increasing in $(C/N, \infty)$. Also note that $\eta(r)$ is continuous at C/N (Eq. 17), and $\eta(C/N) = C/N$. Hence, $\eta(r) \geq C/N$. ☐

When a user sends at r, the best strategy for other users is to send traffic at their Nash rate $\eta(r)$. This is formally stated in the following theorem.

THEOREM 4.3. *If a user (say user 1) sends at r_1 then $\forall_{i \in (2,N)} r_i = \eta(r_1)$ is the unique Nash Equilibrium for the remaining $N-1$ users.*

The proof is provided in the Appendix. Observe that if a user sends at $r_1 \leq C/N$, thereby not contributing to congestion, then the spare capacity is divided equally among the others ($\eta(r_1) = (C - r_1)/(N-1)$). If a user misbehaves and sends traffic at a rate $r_1 > C/N$, while other users remain well behaved, then other users can safely increase their rate upto $\eta(r_1)$, whereas the misbehaving user gets penalized to its residual Nash rate $\eta_R(r)$, defined as follows.

DEFINITION 4. *Given a user with input rate r, define its residual Nash rate as:*

$$\eta_R(r) = C - (N-1)\eta(r) \tag{18}$$

The following two lemmas illustrate that the more a user contributes to congestion the more penalty it incurs.

LEMMA 4.3. *The residual Nash rate is less than or equal to C/N i.e. $\eta_R(r) \leq C/N$.*

This immediately follows from Eq. 18 and Lemma 4.2.

LEMMA 4.4. *$\eta_R(r)$ is strictly decreasing function of r in the range $(C/N, \infty)$.*

The proof immediately follows from Eq. 18 and Lemma 4.1.

A steep weight function results in severer punishment for a user contributing to congestion and larger equilibrium output rate for well behaving users. This is illustrated in Figure 5 which plots Nash rate and residual Nash rate for different diminishing weight functions. As can be easily observed, a steeper weight function($W(r) = 1/r^2$) results in a larger penalty as compared to a less steep weight function ($W(r) = 1/log(r)$).

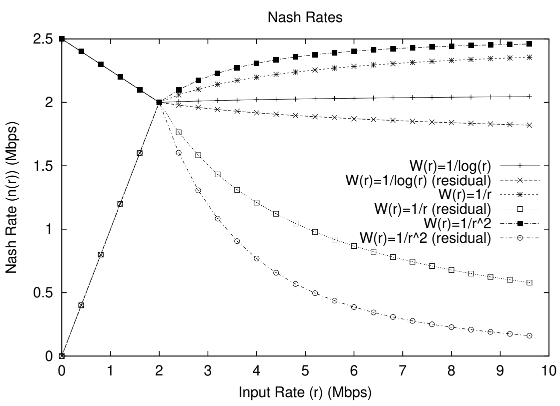

Figure 5: Nash rate

4.2 Arbitrary Network of Links

Consider an arbitrary network servicing N flows. Assume that DWS scheduling is deployed at every link in the network. Assume that the input rate of each flow is constant. Therefore, the input and output rates of users at every other link will also be constant. Now, the input and output rate of each flow at each link can be computed using Eq. 1, 12 and 13.

The following theorem establishes that even in an arbitrary network of links, max-min fair input rates constitute a Nash as well a Stackelberg Equilibrium if DWS schedulers are deployed at each link. Max-min fairness [4] is a well known notion of fairness in an arbitrary network. Denote by $\underline{\mathcal{M}}$, the $1 \times N$ vector of max-min fair rates of these flows through this network.

THEOREM 4.4. *Consider N users sending their traffic as N distinct flows through an arbitrary network with independent DWS scheduling at each link. The max-min fair rates $\underline{\mathcal{M}}$ constitute a Nash as well as a Stackelberg Equilibrium for the users.*

The proof is provided in the Appendix. Furthermore, it is not necessary that the same weight function be used at each link. This makes it easier to adopt DWS in a heterogenous environment with different administrative domains and policies.

Besides $\underline{\mathcal{M}}$, there may be other equilibria also, and users may try to affect which equilibrium to reach. In such a case, it can be shown that at least one user will experience losses in any other Nash Equilibria. This is illustrated in the following Lemma.

LEMMA 4.5. *Consider N users sending their traffic as N distinct flows through an arbitrary network with independent DWS scheduling at each link. The max-min fair rates $\underline{\mathcal{M}}$ constitutes the unique Nash as well as the Stackelberg Equilibrium in which there are no losses in the system.*

The proof is provided in the Appendix. In general, a user's utility may also depend on its loss rate in addition to the output rate. Out of many Nash Equilibrias giving the same output rates, generally users will prefer one with smaller losses. We formally define this class of utility functions ($U^{\gamma l}$) as follows:

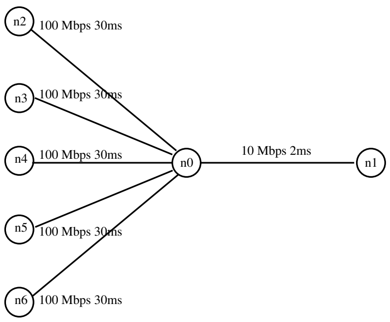

Figure 6: Simulation Scenario

1. $U \in U^{\gamma l}$ maps a user's output rate γ and loss-rate l to a real-valued non-negative utility.
2. $U(\gamma', l) > U(\gamma, l)$ iff $\gamma' > \gamma$.
3. $U(\gamma, l') < U(\gamma, l)$ iff $l' > l$.

If all users have such utility functions, it turns out that $\underline{\mathcal{M}}$ is the unique Nash as well Stackelberg Equilibrium.

This is illustrated in the following theorem which is similar to Theorem 4.1 and Theorem 4.2 for a single link.

THEOREM 4.5. *Consider N users sending their traffic as N distinct flows through an arbitrary network with independent DWS scheduling at each link. Let U_i be the utility function of user i. If $\forall_i U_i \in U^{\gamma l}$, then the max-min fair rates $\underline{\mathcal{M}}$ constitutes the unique Nash and Stackelberg Equilibrium.*

The proof is provided in the Appendix. Therefore the "best selfish behaviour" for a user is to send traffic at its max-min fair rate.

5. SIMULATION RESULTS

The results in the previous section imply that the "best selfish behaviour" for a user in the presence of other similar users is to send traffic at its max-min fair rate. However, the max-min fair rate depends on (a) the link capacities, (b) the number of flows through each link, (c) the input rate of other flows and (d) the path of each flow. A user will not know these parameters in general and thus will not be able to know its max-min fair rate. However, from Lemmas 3.4 and 3.5 it does seem that in case of a single link with DWS scheduling, each iteration of a TCP style increase/decrease algorithm with suitable parameters will bring input rates closer to the fair rates. Therefore, if a single link with DWS scheduling is modeled as a game, then TCP-like end user algorithms seem to be reasonable strategies to play the game.

In this section we illustrate through simulations that the different versions of TCP algorithms are indeed able to converge to their max-min fair rate, (and at Nash rates in the presence of misbehaving users), when DWS schedulers are deployed in the network. The convergence to Nash rates is also shown for different diminishing weight functions. Moreover, specific versions of TCP and the round trip times of individual flows have little impact on the average output rate of a flow. Therefore, DWS scheduling solves most of the problems of congestion control in the presence of misbehaving users [7, 8].

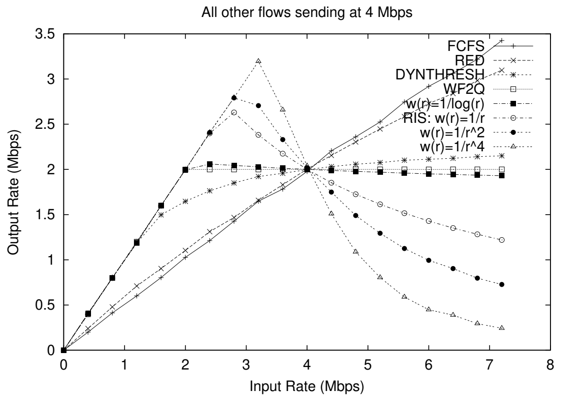

Figure 7: DWS Performance in the presence of CBR flows

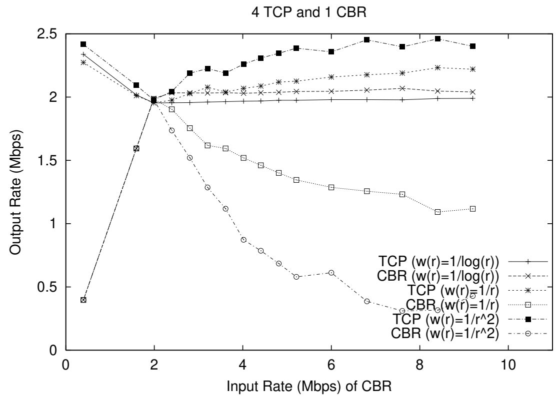

Figure 8: DWS Performance in the presence of TCP and CBR flows

5.1 Simulation Scenario

The simulation scenario is shown in Figure 6. The bottleneck link has a capacity of 10 Mbps and propagation delay of 2 ms. There are five access links of capacity 100 Mbps and propagation delay 30 ms.

The buffer size for a flow at each link was set to its round trip delay- bandwidth product. PDWS with α smoothing, per flow buffers and tail drop was used. NS [1] was used to carry out all the simulations.

5.2 CBR flows

The bottleneck link is shared by five CBR flows, four of which send traffic at a constant rate of 4 Mbps. The rate of the fifth flow is varied from 0 Mbps to 7.2 Mbps. A plot of its output rate vs. the input rate for various scheduling algorithms and buffer management policies is shown in Figure 7.

This is a scenario of heavy congestion. Note from Figure 7, that with DWS scheduling the flow is able to receive its fair rate as long as it does not cause congestion. The flow starts receiving a penalty when its input rate exceeds the fair rate. The amount of penalty depends on the weight function $W()$. Note that the penalties are higher with $W(r) = 1/r^4$ and lower with $W(r) = 1/\log(r)$.

5.3 TCP with unresponsive flows

A flow that does not change its input rate during congestion is referred to as an unresponsive flow [7]. Responsive flows back off by reducing their sending rate upon detecting congestion while unresponsive flows continue to inject packets into the network thereby grabbing a larger share of the bandwidth. As a result the presence of unresponsive flows gives rise to unfairness in bandwidth allocation. With PDWS scheduling deployed, we show that TCP style AIMD algorithms can estimate and send traffic at their max-min fair rate (or Nash rate) even in the presence of misbehaving flows.

The bottleneck link is shared by 5 users, 4 using (responsive) TCP Tahoe and one unresponsive constant bit rate (CBR) source. Responsive TCP flows back off during congestion while unresponsive CBR flows continue to inject packets into the network thus attempting to grab a larger share of the bandwidth. Figure 8 shows the average output rates for a representative TCP flow as the input rate of the CBR flow is varied. Each point in the graph represents a simulation of 20 seconds. However, the output rates correspond to the average rate in the last 10 seconds of the simulation when they get stabilized.

Figure 8 shows that TCP flows are able to get close to their Nash rate (shown in Figure 5) and hence to their Nash Equilibrium (according to Theorem 4.3).

Note that with $W(r) = 1/\log(r)$ the output rate of CBR flow is greater than that of the TCP flow. This is because the inverse log weight function gives very little penalty to the misbehaving CBR flow and is very similar to WFQ. The bandwidth left by TCP because of timeouts and retransmits is grabbed by the CBR flow despite its (slightly) small GPS weight. As CBR increases its rate further, the penalty slowly increases allowing TCP to grab a larger share.

5.4 TCP Versions

Different versions of TCP like Tahoe, Reno [21], Vegas [5], and Sack [20] are known to perform differently [21]. We show that with DWS scheduling there is very little difference in the output rates achieved by these versions. The simulation scenario is shown in Figure 6 with different versions of TCP at n2, n3, n4 and n5. Packetized rate inverse scheduler (PRIS) was used at the bottleneck link. Figure 9 shows the total bytes of a flow transferred as a function of time. The output rate is given by the slope of the graph. Since the slopes are almost identical we see that all versions get identical rates.

5.5 Multiple vs Single Connection

Opening multiple simultaneous connections is a very simple way to grab more bandwidth from simple FCFS (droptail) gateways. The simulation scenario is shown in Figure 6 with FCFS employed at node n0. The users at nodes n2, n3, n4, n5 and n6 open 1, 2, 3, 4 and 5 TCP Reno connections to node n1 respectively.

Typically, a user opening more simultaneous connection is able to grab more bandwidth. However, this is not the case with DWS scheduling when all the TCP connections of a user are treated as a single flow. Figure 10 shows a plot of bytes transferred vs. time when PRIS is deployed at node n0. We see that all users get an almost identical bandwidth.

5.6 TCP with different Round Trip Times

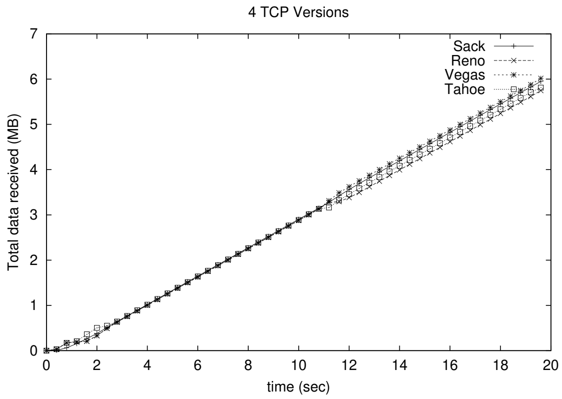

Figure 9: PRIS Performance in the presence of different TCP versions

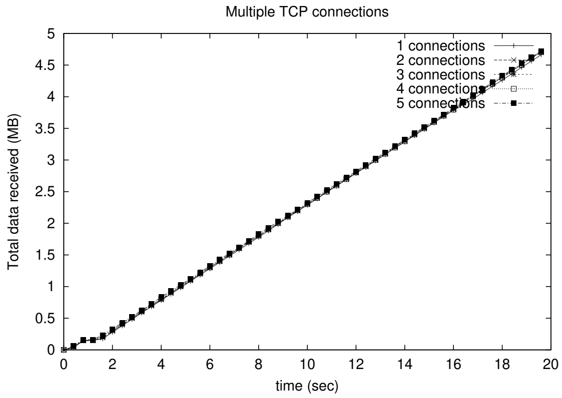

Figure 10: PRIS Performance in the presence of multiple connections

The Round Trip Time (RTT) of a TCP connection determines how fast it adapts itself to the current state of the network. A connection with smaller RTT is able to infer the status of the network earlier than a connection with larger RTT. Therefore, large RTT connections typically achieve lower output rates.

The simulation scenario is the same as shown in Figure 6 except that the propagation delays of links (n2-n0), (n3-n0), (n4-n0), (n5-n0) and (n6-n0) are set to 5, 10, 20, 50, 100 ms respectively. PRIS scheduler was used and the value of α was taken to be 0.9 which corresponds to the α of the minimum RTT flow according to Eq 16. Figure 11 shows that when PRIS is deployed, after a few transients initially, all flows are able to achieve almost identical rates.

5.7 Network

In this section we show that with DWS scheduling deployed in a network adaptive flows like TCP are able to estimate and converge to their max-min fair rates. The simulation scenario is shown in Figure 12. There are 6 TCP Reno flows. The paths of flows 0, 1, 2, 3, 4, and 5 are (n0-n2-n1), (n4-n3-n5), (n0-n2-n3-n4), (n1-n2-n3-n5), (n0-n2-n3-n5) and (n1-n2-n3-n4) respectively. The buffer size of each flow on each gateway was taken to be 300 packets which is the bandwidth delay product of the flow with the

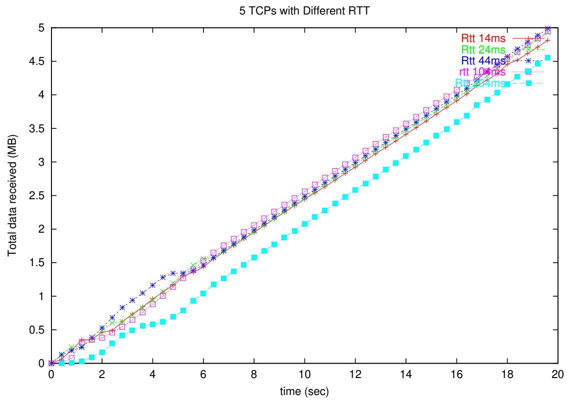

Figure 11: PRIS Performance in the presence flows with varying RTTs

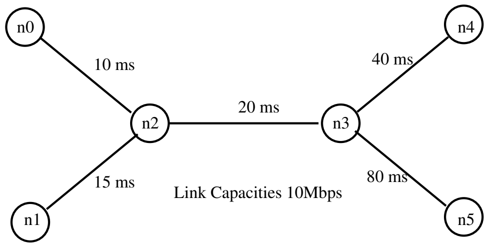

Figure 12: Simulation scenario for a network

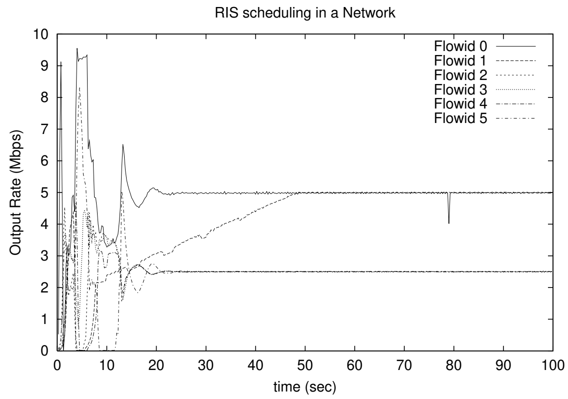

Figure 13: Output rates of flows in a network with PRIS

largest RTT. For this scenario the max-min fair rate [4] for flows 0 and 1 is 5 Mbps and for flows 2, 3, 4, and 5 is 2.5 Mbps.

Figure 13 shows a plot of output rate vs. time for all 6 flows. We see that after some initial transients all flows converge to their max-min fair rates.

6. CONCLUSIONS

Using the techniques of game-theory, we showed that the current resource sharing mechanisms in the Internet either encourage congestion causing behaviour, or are oblivious to it. While these mechanisms may be adequate currently, their applicability in the future remains questionable. With growth, heterogeneity and commercialization of the Internet, the assumption of end-users being cooperative might not remain valid. This may lead to a congestion collapse of the Internet due to selfish behaviour of the end-users.

We proposed a class of switch scheduling algorithms by the name Diminishing Weight Schedulers (DWS) and showed that they encourage congestion avoiding behaviour and punish behaviours that lead to congestion. We showed that for a single link with DWS scheduling, fair rates constitute the unique Nash and Stackelberg Equilibrium. We also showed that for an arbitrary network with DWS scheduling at every link, the max-min fair rates constitute a Nash as well as a Stackelberg Equilibrium. Therefore, when DWS schedulers are deployed, even the selfish users will try to estimate their max-min fair rate and send traffic only at that rate.

It is possible to set different DWS weights for different users (or traffic classes). This should lead to (in a game-theoretic manner) weighted fair sharing in case of a single link and weighted max-min fair sharing in case of a network. These weights may be set in accordance with the pricing or other resource sharing policies.

We defined the concept of Nash rate and showed how the choice of different weight functions can affect the reward-penalty profile of DWS. With the $1/r^2$ diminishing weight function, the penalty imposed is large, whereas with $1/\log(r)$ diminishing weight function, the behaviour of DWS is only marginally different from that of GPS which imposes no penalty. Therefore DWS may also be viewed as a generalization of GPS scheduling with suitable game-theoretic properties. DWS does not require different nodes to use the same weight function. Therefore, it is well suited for heterogenous environment consisting of different administrative domains, where each domain may independently choose a diminishing weight function according to its administrative policies.

Although the max-min rate constitute Nash and Stackelberg Equilibrium, it is not clear how users can estimate their max-min fair rates. For this, a decentralized distributed scheme such as the one is proposed [15] is required. Moreover one needs to establish that such a distributed scheme will be stable and will indeed converge to the max-min fair rates when DWS schedulers are deployed in the network. This is a topic under investigation. Our current paper does not address this issue in a theoretical framework. Also, in this paper we assumed that the input rate of every user is constant. With a distributed scheme to estimate the max-min fair rate, this assumption will not remain valid. Analyzing DWS scheduling with dynamically changing rates is another open problem.

We believe that, it should be possible to design distributed algorithms that are stable and converge to the max-min fair rates in presence of DWS scheduling. It seems that additive increase and multiplicative decrease algorithms (such as the one followed by TCP) with proper engineering may perform well with DWS scheduling.

Using simulations we showed that in a network with DWS scheduling, most of the TCP variants are able to estimate their max-min fair rate reasonably well, irrespective of their versions and round trip times (RTT). We also showed that with DWS, the TCP users indeed get rewarded according to their Nash rates in the presence of unresponsive, misbehaving CBR flows which get punished.

Our proposed model requires per-flow queueing and scheduling in the core routers, which may not be very easy to implement in a realistic situation. However, this work presents a significantly different view of resource sharing and congestion control in communication networks and gives a a class of scheduling algorithms that can be used to solve the problem in a game-theoretic framework. Based on this work, one may be able to design "core stateless" policies [29] with similar properties.

7. ACKNOWLEDGEMENTS

We thank the computer communication review editors and the anonymous reviewers for their helpful comments on this paper.

8. REFERENCES

[1] Network simulator (NS), 2001. URL: http://www.isi.edu/nsnam/ns/.

[2] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. In *IEEE Symposium on Foundations of Computer Science*, pages 604–612, 1994.

[3] J. C. R. Bennett and H. Zhang. WF2Q: worst-case fair weighted fair queueing. In *Proceedings of INFOCOM*, pages 120–128, San Fransisco, California, Mar. 1996.

[4] D. P. Bertsekas and R. G. Gallager. *Data Networks.* Prentice Hall, Englewood Cliffs, NJ, USA, 1992.

[5] L. S. Brakmo and S. W. O'Malley. TCP Vegas: New techniques for congestion detection and avoidance. In *Proceedings of SIGCOMM*, pages 34–35, London, United Kingdom, Aug. 1994. ACM.

[6] A. K. Choudhury and E. L. Hahne. Dynamic queue length thresholds in a shared memory ATM switch. In *Proceedings of INFOCOM*, San Fransisco, California, Mar. 1996.

[7] S. Floyd. Congestion control principles. Request for Comments 2914, Internet Engineering Task Force, Sept. 2000.

[8] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[9] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. Request for Comments 2582, Internet Engineering Task Force, Apr. 1999.

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

[11] D. Fudenberg and J. Tirole. *Game Theory.* MIT Press, October 1991.

[12] R. Garg, A. Kamra, and V. Khurana. Eliciting cooperation from selfish users: A game-theoretic approach towards congestion control in communication networks. Technical Report RI01001, IBM Research, April 2001. Available at http://www.research.ibm.com/resources/paper_search.html.

[13] S. J. Golestani. A self-clocked fair queuing scheme for broadband applications. In *Proceedings of INFOCOM*, pages 636–646, April 1994.

[14] V. Jacobson. Congestion avoidance and control. *Computer Communications Review*, 18(4):314–329, Aug. 1988.

[15] J. M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, 29(7):954–962, 1981.

[16] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan. ERICA switch algorithm: A complete description. Technical report, ATM Forum/96-1172, August 1996.

[17] H. T. Kung, T. Blackwell, and A. Chapman. Credit update protocol for flow-controlled ATM networks: Statistical multiplexing and adaptive credit allocation. In *Proceedings of SIGCOMM*, pages 101–114, London, UK, Sept. 1994.

[18] D. Lin and R. Morris. Dynamics of random early detection. In *Proceedings of the ACM SIGCOMM'97 Conference*, pages 127–138, New York, September 1997.

[19] M. Mathis and J. Mahdavi. Forward acknowledgement: Refining TCP congestion control. *Computer Communications Review*, 26(4):281–291, Oct. 1996.

[20] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. Request for Comments 2018, Internet Engineering Task Force, Oct. 1996.

[21] J. Mo, R. La, V. Anantharam, and J. Walrand. Analysis and comparison of TCP Reno and Vegas. In *Proceedings of INFOCOM*, New York, Mar. 1999.

[22] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control - the single node case. In *Proceedings of INFOCOM*, May 1992.

[23] J. Postel. Transmission control protocol. Request for Comments 793, Internet Engineering Task Force, Sept. 1981.

[24] W. Prue and J. Postel. Something a host could do with source quench: The source quench introduced delay (SQuID). Request for Comments 1016, Internet Engineering Task Force, July 1987.

[25] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. Request for Comments 2481, Internet Engineering Task Force, Jan. 1999.

[26] K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks,. *ACM Trans. on Computer Systems*, 8(2):158–181, May 1990.

[27] S. Shenker. Making greed work in networks: A game-theoretic analysis of switch service disciplines. In *Proceedings of SIGCOMM*, pages 47–57, London, UK, Sept. 1994.

[28] D. Stiliadis and A. Varma. Design and analysis of frame-based fair queuing: A new traffic scheduling algorithm for packet switched networks. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 104–115, New York, May 1996.

[29] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *ACM Sigcomm '98*, Septermber 1998.

[30] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhury. Buffer management schemes for supporting TCP in gigabit routers with per-flow queueing. *IEEE Journal on Selected Areas in Communications*, 17(6):1159–1169, June 1999.

[31] A. Varma and D. Stiliadis. Hardware implementation of fair queueing algorithms for asynchronous transfer mode networks. *IEEE Communications Magzine*, pages 54–68, December 1997.

[32] C.-Q. Yang and A. V. S. Reddy. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network Magazine*, 9(5), July/August 1995.

[33] H. Zhang and J. C. R. Bennett. Hierarchical packet fair queueing algorithms. In *Proceedings of ACM SIGCOMM*, pages 143–156. ACM, September 1997.

[34] L. Zhang. Virtual clock : A new traffic control algorithm for packet switching networks. *ACM Transactions on Computer Systems*, 9:101–124, May 1991.

9. APPENDIX

Before proceeding to the proofs of our main theorems, we prove certain properties.

LEMMA 9.1. *The congestion characteristic function $G(x, \underline{r})$, as defined in Eq. 10, is a strictly increasing function of x in $(0, k_{max})$ and is constant in $[k_{max}, \infty)$, where $k_{max} = \max_{0 \le i \le N}\{r_i/W(r_i)\ \}$.*

PROOF. Note that

$$x < x' \Rightarrow, \forall j : \min(x.W(r_j), r_j) \le \min(x'.W(r_j), r_j). \quad (19)$$

Therefore, $G(x, \underline{r}) \le G(x', \underline{r})$.

Consider $x \in (0, k_{max})$, since $0 < x < k_{max}$, $\exists i$ s.t. $x < r_i/W(r_i)$ by definition of k_{max}. Therefore, $\min(x.W(r_i), r_i) = x.W(r_i)$. Hence for $x < x' < k_{max}$, we get $\min(x.W(r_i), r_i) < \min(x'.W(r_i), r_i)$. For all other flows j Eq. 19 holds. Hence summing over all flows we get $G(x, \underline{r}) < G(x', \underline{r})$.

Now consider $x \in [k_{max}, \infty)$, since $x \ge k_{max}$ we get $\forall i$ $x \ge \frac{r_i}{W(r_i)}$. Therefore, $\forall i$ $\min(x.W(r_i), r_i) = r_i$. The same holds for any $x' > x$. Hence we get, $\forall i$ $\min(x.W(r_i), r_i) = \min(x'.W(r_i), r_i) = r_i$. Summing over all flows we get $G(x, \underline{r}) = G(x', \underline{r}) = \sum_{i=1}^{N} r_i$ □

PROOF. Theorem 3.1: **[Uniqueness of κ]** If $\sum_{i=1}^{N} r_i \le C$, then κ is unique by definition. Using Lemma 9.1, $G(x, \underline{r})$

is a monotonically increasing function of x in the domain $(0, k_{max})$. Now if $\sum_{i=1}^{N} r_i > C$ then $G(k_{max}, \underline{r}) = \sum_i r_i > C$. Since κ satisfies $G(x, \underline{r}) = C$, using Lemma 9.1, $\kappa < k_{max}$. Therefore, κ is unique.

[Output rate] If $\sum_{i=1}^{N} r_i \le C$, from the flow conservation and the work conservation properties of DWS scheduler, $\gamma_i = r_i \le CW(r_i)/W(C)$, (since $r_i \le C$ and $W()$ is strictly decreasing). Since $\kappa = C/W(C)$, it follows that $\gamma_i = \min(\kappa.W(r_i), r_i)$.

If $\sum_{i=1}^{N} r_i > C$, then there must exist at least one flow i such that $\gamma_i < r_i$. Flow i will have a non-zero backlog. From the DWS fairness property, Eq. 9, we have $\gamma_i = k.W(r_i) < r_i$. It follows that

$$\gamma_i = \min(k.W(r_i), r_i). \tag{20}$$

Now consider a flow j such that $\gamma_j = r_j$. From DWS fairness, Eq. 8, we get $\gamma_j/W(r_j) \le k$. Therefore, $\gamma_j = r_j \le k.W(r_j)$. It follows that

$$\gamma_j = \min(k.W(r_j), r_j). \tag{21}$$

Since the scheduler has a positive backlog, from the flow conservation and work conservation properties of DWS, $\sum_{i=1}^{N} \gamma_i = C$. Substituting values of γ_is of backlogged flows from Eq. 20 and γ_is of non-backlogged flows from Eq. 21, we get :

$$\sum_{i=1}^{N} \min(k.W(r_i), r_i) = C.$$

Since the rate constant κ is unique value satisfying the equation $G(x, \underline{r}) = C$, we get $k = \kappa$. Therefore: $\gamma_i = \min(\kappa.W(r_i), r_i)$ for all flows i. □

Proof. Lemma 3.1: From Theorem 3.1, κ is unique. We show the uniqueness of f using contradiction. Suppose f is not unique then there are 2 distinct points x_1 and x_2 which satisfy $x = \kappa.W(x)$. Since x_1 and x_2 are distinct, WLOG assume that $x_1 < x_2$. Since $x_1 = \kappa.W(x_1)$ and $x_2 = \kappa.W(x_2)$ we get $W(x_1) < W(x_2)$ (note κ is always positive). However, since $W()$ is strictly decreasing $x_1 < x_2 \Rightarrow W(x_1) > W(x_2)$ which leads to a contradiction. Hence f is unique. □

Proof. Lemma 3.3: If $\sum_{i=1}^{N} r_i \le C$ then $\kappa = C/W(C)$. Using Eq. 12 we get $f/W(f) = C/W(C)$. Hence $f = C \ge C/N$.

If $\sum_{i=1}^{N} r_i > C$, then from Corollary 3.1, $f \ge \gamma_i$. Summing over all flows we get $fN \ge \sum_{i=1}^{N} \gamma_i$. But from the flow conservation and work conservation properties of DWS scheduler $\sum_{i=1}^{N} \gamma_i = C$, therefore $f \ge C/N$. □

Proof. Lemma 3.4: WLOG assume that flow 1 is experiencing losses, i.e. $r_1 > f > \gamma_1$, where f is the fair rate which satisfies $f = \kappa W(f)$. Now $\sum_i r_i > C$, otherwise $\forall_i \gamma_i = r_i$. The rate constant κ satisfies $G(\kappa, \underline{r}) = C$. Now let $\delta = r_1 - f$. Then the new input rate of flow 1 is $r_1' = r_1 - \delta = f$. Let the new rate vector be $\underline{r}'$.

Consider $G(\kappa, \underline{r}')$. Since $min(r_1', \kappa W(r_1') = min(f, f) = f > \gamma_1$, it follows that $G(\kappa, \underline{r}') > G(\kappa, \underline{r}) = C$. Using Lemma 9.1, $G(x, \underline{r})$ is an increasing function of x, it follows that $\kappa' < \kappa$.

Now $\forall_2^N \gamma_i' = min(r_i, \kappa' W(r_i)) \le min(r_i, \kappa W(r_i)) = \gamma_i$, since $\kappa' < \kappa$. Now $\gamma_1' = C - \sum_2^N min(r_i, \kappa' W(r_i)) \ge C - \sum_2^N min(r_i, \kappa W(r_i)) = \gamma_1$. □

Proof. Lemma 3.5: WLOG assume that flow 1 is not experiencing losses, i.e. $r_1 = \gamma_1 < f$, where f is the fair rate which satisfies $f = \kappa W(f)$. The rate constant κ satisfies $G(\kappa, \underline{r}) = C$. Three cases arise:

Case 1: $\sum_i r_i < C$: Flow r_1 can increase its input rate by an amount $C - \sum_i r_i$ and get a better output.

Case 2: $\sum_i r_i = C$: In this case, $\forall_i \gamma_i = r_i$. Now let $r_1' = r_1 + \delta$. The other flows do not change their input rates, i.e. $\forall_i r_i$ remains unchanged. Then $\gamma_1' = C - \sum_{i=2..N} \gamma_i' \ge C - \sum_{i=2..N} r_i = \gamma_1$, since $\forall_{i=2..N} \gamma_i' \le r_i$. Hence the output of flow 1 will may increase or remain the same.

Case 3: $\sum_i r_i > C$: Now the rate constant κ satisfies $G(\kappa, \underline{r}) = C$. Now let $\delta = f - r_1$. Then the new input rate of flow 1 is $r_1' = r_1 + \delta = f$. Let the new rate vector be $\underline{r}'$.
Consider $G(\kappa, \underline{r}')$. Since $min(r_1', \kappa W(r_1') = min(f, f)$ $= f > \gamma_1$, it follows that $G(\kappa, \underline{r}') > G(\kappa, \underline{r}) = C$. Using Lemma 9.1, $G(x, \underline{r})$ is an increasing function of x, it follows that $\kappa' < \kappa$.
Now $\forall_2^N \gamma_i' = min(r_i, \kappa' W(r_i)) \le min(r_i, \kappa W(r_i)) = \gamma_i$, since $\kappa' < \kappa$. Now $\gamma_1' = C - \sum_2^N min(r_i, \kappa' W(r_i)) \ge C - \sum_2^N min(r_i, \kappa W(r_i)) = \gamma_1$.

This completes the proof. □

Lemma 9.2. *If the input rate of flow j is decreased to r_j' from r_j ($r_j' < r_j$), then either $\gamma_j' \ge \gamma_j$ or $\gamma_j' = r_j'$.*

Proof. If $\sum_{i=1}^{N} r_i \le C$, then we have $\forall_i \gamma_i = r_i$, from the flow conservation and work conservation properties of the DWS scheduler. If r_j is decreased to r_j', then $\sum_{i=1(i \ne j)}^{N} r_i + r_j' \le C$ still holds. Therefore, $\gamma_j' = r_j'$ even now.

We now consider the case when $\sum_{i=1}^{N} r_i > C$. Again from the flow conservation and work conservation properties of the DWS scheduler, we have $\sum_{i=1}^{N} \gamma_i = C$. Using Lemma 3.2 we get $\sum_m r_m + \sum_n \kappa.W(r_n) = C$ where for all flows m, $r_m \le f$ and for all flows n, $r_n > f$. Therefore,

$$\kappa = \frac{C - \sum_m r_m}{\sum_n W(r_n)} \tag{22}$$

We see that the value of κ also changes to κ' on changing the value of r_j. We have the following cases:

Case 1 ($r_j \le f$): Since $r_j \le f$, j is one among the m flows. We rewrite Eq. 22 in the form

$$\kappa = \frac{C - \sum_{m(m \ne j)} r_m - r_j}{\sum_n W(r_n)} \tag{23}$$

Hence we see that κ increases on decreasing r_j, and hence f also increases (using Eq. 12). Therefore, $r_j' < r_j \le f < f'$. Since $r_j' < f'$, using Lemma 3.2 we get $\gamma_j' = r_j'$.

Case 2 ($r_j > f$): Since $r_j > f$, j is one among the n flows. We rewrite Eq. 22 in the form

$$\kappa = \frac{C - \sum_m r_m}{\sum_{n(n \ne j)} W(r_n) + W(r_j)} \tag{24}$$

We again have two cases, if $r'_j \leq f'$, then using Lemma 3.2 $\gamma'_j = r'_j$. If $r'_j > f'$, we show that $\gamma'_j > \gamma_j$. Define the residual capacity by $C_{res} = C - \sum_m r_m$. Since $r_j > f$ and $r'_j > f'$ we have:

$$\gamma_j = \kappa.W(r_j) = \frac{C_{res}}{(\frac{1}{W(r_j)}\sum_{n(n\neq j)} W(r_n) + 1)} \qquad (25)$$

$$\gamma'_j = \kappa'.W(r'_j) = \frac{C_{res}}{(\frac{1}{W(r'_j)}\sum_{n(n\neq j)} W(r_n) + 1)} \qquad (26)$$

using Eq. 24. Since $r_j > r'_j$ and $W()$ is a strictly decreasing function, we get $\gamma'_j > \gamma_j$. □

LEMMA 9.3 (FLOW REMOVAL). *If $\sum_{i=1,i\neq j}^{N} r_i > C - \gamma_j$ then removal of flow j and adjustment of link capacity as $C' = C - \gamma_j$ does not change the rate constant κ or the fair rate f.*

PROOF. From Theorem 3.1, rate constant κ satisfies $G(\kappa, \underline{r}) = C$. Rewrite the above equation in the form

$$\sum_{i=1(i\neq j)}^{N} \min(\kappa.W(r_i), r_i) = C - \gamma_j \qquad (27)$$

Substitute the new capacity C' and the new rate vector $\underline{r}'$ in the above equation to get $G(\kappa, \underline{r}') = C'$. Since $\sum_{i=1,i\neq j}^{N} r_i > C'$, and κ is the unique rate satisfying $G(\kappa, \underline{r}') = C'$, the rate constant κ remains unaffected by removing flow j.

Since the rate constant κ is unaffected, the fair rate f also remains unchanged. □

LEMMA 9.4. *If $r > C/N$ then: (1). $\eta(r) < r$ and (2). $\forall \underline{r} : r_1 = r, f(\underline{r}) \geq \eta(r) > C/N$.*

PROOF. If $r > C/N$ then $\eta(r)$ satisfies:

$$\frac{xW(r)}{W(x)} + (N-1)x = C \qquad (28)$$

Since LHS is a strictly increasing function of x, it is easy to see that there is exactly one positive real root of the above equation.

1. Suppose $\eta(r) \geq r$. Then LHS of equation 28 becomes $\frac{\eta(r)W(r)}{W(\eta(r))} + (N-1)\eta(r) \geq Nr > C$ since $W()$ is a decreasing function and $r > C/N$. So equation 28 can never be satisfied. Hence $\eta(r) < r$.
2. Consider an arbitrary rate $r' \geq \eta(r)$. Since $W()$ is a diminishing weight function, $\frac{W(r')}{W(\eta(r))} \leq 1$. Therefore, $\eta(r)\frac{W(r')}{W(\eta(r))} \leq \eta(r) \leq r'$ and hence: $min(r', \eta(r)\frac{W(r')}{W(\eta(r))}) \leq \eta(r)$.
   Now consider an $r' < \eta(r)$. Clearly, $min(r', \eta(r)\frac{W(r')}{W(\eta(r))}) = r' \leq \eta(r)$. Therefore, for any rate r',

$$min(r', \eta(r)\frac{W(r')}{W(\eta(r))}) \leq \eta(r) \qquad (29)$$

   Also,

$$min(r_1, \eta(r)\frac{W(r_1)}{W(\eta(r))}) = \eta(r)\frac{W(r)}{W(\eta(r))} \qquad (30)$$

   Substituting $r' = r_i$ in Eq. 29 and summing over $i = 2..N$ and adding Eq. 30, we get: $\sum_{i=1}^{N} min(r_i, \eta(r)\frac{W(r_i)}{W(\eta(r))}) \leq \eta(r)\frac{W(r)}{W(\eta(r))} + (N-1)\eta(r)$. Substituting from Eqs. 10 and 17, we get $G(\frac{\eta(r)}{W(\eta(r))}, \underline{r}) \leq C$. From the above equation and from the definition of κ in Eq. 11, and the fact that $G(x, \underline{r})$ is an increasing function of x, it is easy to see that $\kappa \geq \frac{\eta(r)}{W(\eta(r))}$. From Eq. 12, $f = \kappa W(f)$ and hence $\frac{f}{W(f)} \geq \frac{\eta(r)}{W(\eta(r))}$.
   Since $W()$ is a monotonically decreasing function, no $f < \eta(r)$ can satisfy the above equation. Hence $f \geq \eta(r)$.

This completes the proof. □

PROOF. Theorem 4.1: **[Nash Equilibrium]** Let $\forall_i r_i = C/N$. Then $\forall_i \gamma_i = C/N$ since $f \geq C/N$ always. WLOG, suppose that user 1 changes its input rate to r'_1.

Case 1: $\mathbf{r'_1 < C/N}$: Clearly, $\gamma_1 < C/N$.

Case 2: $\mathbf{r'_1 > C/N}$: Now $\sum_{i=1}^{N} r_i > C$, so the scheduler will have a positive backlog. From Lemma 3.3, $f \geq C/N$. Now $\forall_{i\in(2..N)} r_i = C/N$, so from Lemma 3.2, $\forall_{i\in(2..N)}\gamma_i = C/N$. So, from the flow conservation and work conservation properties of DWS $\gamma_1 = C - \sum_{i\in(2..N)} \gamma_i = C/N$.

So user 1 can never increase its output rate by unilaterally changing its input rate. Hence $\forall_i r_i = C/N$ is a Nash Equilibrium.

[Uniqueness] Assume WLOG a vector for input rates $\underline{r}$ constituting a Nash Equilibrium, s.t $r_1 \neq C/N$. There are two cases:

Case 1: $\mathbf{\exists_i r_i < C/N}$: Here $\gamma_i < C/N$. So by increasing r_i to C/N, γ_i can be increased to C/N. Therefore $\underline{r}$ does not constitute a Nash Equilibrium.

Case 2: $\mathbf{\forall_i r_i \geq C/N}$ and $\mathbf{r_1 > C/N}$: WLOG, assume that $r_1 = max(r_1, r_2, ...r_N) > C/N$. There are three sub-cases:

(a) $\mathbf{\exists_{i\geq 2}\gamma_i < \eta(r_1)}$: Using Lemma 9.4, $\forall_{r_2,r_3...r_N} f \geq \eta(r_1)$. So by making $r_i = \eta(r_1)$, γ_i can be increased to $\eta(r_1)$ (Lemma 3.2).

(b) $\mathbf{\forall_{i\geq 2}\gamma_i = \eta(r_1)}$: $\gamma_1 = C - \sum_{i\in(2,N)} \gamma_i = C - (N-1)\eta(r_1) < C/N$. So by making $r_1 = C/N$, γ_1 can be increased to C/N.

(c) $\mathbf{\forall_{i\geq 2}\gamma_i \geq \eta(r_1)}$ and $\mathbf{\exists_{j\geq 2}\gamma_j > \eta(r_1)}$: It is easy to see that $f < r_{max} = r_1$. From Lemma 9.4 $f \geq \eta(r_1)$. Now $f = \kappa W(f)$ and since $\eta(r_1) \leq f$, it follows that $\eta(r_1) < \kappa W(\eta(r_1))$. So $\gamma_1 = \kappa W(r_1) \geq \frac{\eta(r_1)W(r_1)}{W(\eta(r_1))}$. Now $\sum_i \gamma_i = \gamma_1 + \gamma_j + \sum_{i\geq 2, i\neq j} \gamma_i > \frac{\eta(r_1)W(r_1)}{W(\eta(r_1))} + \eta(r_1) + (N-2)\eta(r_1) = C$. So this case is not possible.

It follows that $\forall_i r_i = C/N$ is a unique Nash Equilibrium. □

PROOF. Theorem 4.3: There are two cases:

Case 1: $\mathbf{r_1 \leq C/N}$: In this case $\gamma_1 = r_1$, irrespective of the value of r_1 because $f \geq C/N$. From Lemma 9.3, removing flow 1 will neither change the rate constant κ nor the fair rate f. The spare capacity $C_{res} = C - r_1$, will be divided among the remaining users. Using Theorem 4.1, the unique Nash Equilibrium for the remaining users is when each sends at $C_{res}/(N-1)$ which is $\eta(r_1)$.

Case 2: $\mathbf{r_1 > C/N}$: Let $\forall_{i\in(2,N)} r_i = \eta(r_1)$. We now show that this vector of input rates constitute a Nash Equilibrium for the remaining users. Using Lemma 9.4, $f \geq \eta(r_1)$ so $\gamma_{i\in(2,N)} = \eta(r_1)$. WLOG, assume that user 2 deviates its rate from $r_2 = \eta(r_1)$ to r_2'. There are the following two sub-cases:

(a) $\mathbf{r_2' < \eta(r_1)}$: Here $\gamma_2' = r_2 < \eta(r_1)$. So clearly this cannot be a Nash Equilibrium.

(b) $\mathbf{r_2' > \eta(r_1)}$: If f' is the new fair rate, then $f' \geq \eta(r_1)$. Since $W(x, \underline{r})$ is a decreasing function of x, we get $\kappa' = \frac{f'}{W(f')} \geq \frac{\eta(r_1)}{W(\eta(r_1))}$. Now using Eq. 30, $\gamma_1' = \min(r_1, \kappa' W(r_1)) = \kappa' W(r_1) \geq \frac{\eta(r_1)W(r_1)}{W(\eta(r_1))}$ and so $\gamma_2' = C - \sum_{i\neq 2}\gamma_i' \leq C - (\frac{\eta(r_1)W(r_1)}{W(\eta(r_1))} + (N-2)\eta(r_1)) = \eta(r_1)$. So the output rate of user 2 cannot increase when it deviates from $\underline{r}$. Hence $\underline{r}$ is a Nash Equilibrium for the N - 1 user subsystem.

We now show that the above Nash Equilibrium is unique. Consider any other input rate profile $\underline{r}'$. Let $\underline{\gamma}'$ be the corresponding output rate profile. There are two sub-cases:

(a) $\mathbf{\sum_{1=1}^{N} \gamma_i' < C}$: Clearly $\exists_j$ s.t. $\gamma_j' = r_j' < C/N$. Now, r_j' can be increased to C/N to get an output rate of C/N. So this cannot be a Nash Equilibrium.

(b) $\mathbf{\sum_{i=1}^{N} \gamma_i = C}$ and $\mathbf{\exists_{i\in(2,N)} r_i \neq \eta(r_1)}$: Since $f \geq \eta(r_1)$ and $W(x, \underline{r})$ is an increasing function of x, we get $\kappa = \frac{f}{W(f)} \geq \frac{\eta(r_1)}{W(\eta(r_1))}$. Using Eq. 30, we get $\gamma_1 = \frac{f}{W(f)}W(r_1) \geq \frac{\eta(r_1)}{W(\eta(r_1))}W(r_1)$. From the flow conservation and work conservation properties, $\sum_{i\in(2,N)}\gamma_i = C - \gamma_1 \leq C - \frac{\eta(r_1)}{W(\eta(r_1))}W(r_1) = (N-1)\eta(r_1)$. $\Rightarrow \exists_j \gamma_j \leq$ '$\eta(r_1)$. Since by making $r_j = \eta(r_1)$, γ_j can be made equal to $\eta(r_1)$, this can never be a Nash Equilibrium for the N-1 user subsystem.

Hence if user 1 sends at r_1, then $\forall_{i\in(2,N)} r_i = \eta(r_1)$ is the unique Nash Equilibrium for the remaining N-1 users. □

PROOF. Theorem 4.2: WLOG assume that user 1 becomes the leader and sends at r_1 in a Stackelberg Equilibrium.

Case 1: $\mathbf{r_1 < C/N}$ and $\mathbf{\gamma_1 < C/N}$: This cannot be the case since $f \geq C/N$ and making $r_1' = C/N$ makes $\gamma_1' = C/N > \gamma_1$.

Case 2: $\mathbf{r_1 > C/N}$: Using Theorem 4.3, the unique Nash Equilibrium for the remaining flows is when the input and output rate of each of the N-1 users is equal to $\eta(r_1)$. From Lemma 9.4 $\eta(r_1) > C/N \Rightarrow \gamma_1 \leq C - \sum_{i\in(2,N)}\gamma_i = C - (N-1)\eta(r_1) < C - (N-1)C/N = C/N$. This cannot be a Stackelberg Equilibrium as making $r_1' = C/N$ makes $\gamma_1' = C/N > \gamma_1$. The only case left is the following.

Case 3: $\mathbf{r_1 = C/N}$: $\gamma_1 = C/N$ since $f \geq C/N$ always.

So every Stackelberg Equilibrium will have $r_1 = C/N$. Since the unique Nash Equilibrium for other flows is $\forall_{i\in(2,N)} r_i = \eta(r_1) = C/N$, it follows that the only Stackelberg Equilibrium is when $\forall_i r_i = C/N$. □

Before proving Lemma 4.5 we give a brief overview of max-min fair rate computation in an arbitrary network. Let C_ℓ denote the capacity of link ℓ and N_ℓ denote the number of flows passing through link ℓ. Let F_ℓ be the set of flows passing through link ℓ.

The algorithm to compute max-min fair rates [4] works as follows. At every stage k (k is initially 1), it finds the minimum bottleneck link ℓ s.t. for all other links ℓ', $C_\ell^k/N_\ell^k \leq C_{\ell'}^k/N_{\ell'}^k$. WLOG, it labels this link as ℓ_k. For all residual flows P^k passing through ℓ_k (k = 1), it assigns max-min fair rates $\mathcal{M}_{i\in P^k} = C_\ell^k/N_\ell^k$, where C_ℓ^k and N_ℓ^k are respectively the residual capacity and the residual number of flows passing through the link ℓ_k. The link ℓ_k and all the flows in P^k are removed and the residual capacities and residual number of flows of all the other links are updated as follows:

$$C_\ell^{k+1} = C_\ell^k - \sum_{i\in P^k, i\in F_\ell} \mathcal{M}_i \tag{31}$$

$$N_\ell^{k+1} = N_\ell^k - \sum_{i\in P^k, i\in F_\ell} 1 \tag{32}$$

Initially, $C_\ell^1 = C_\ell$ and $N_\ell^1 = N_\ell$. The algorithm terminates when there are no more flows to be removed.

PROOF. Lemma 4.5: Consider the set of flows F_{ℓ_1} through the minimum bottleneck link ℓ_1.

CLAIM 1. *In any Nash or Stackelberg Equilibrium without losses, flows $i \in F_{\ell_1}$ will have $r_i = \gamma_i = C_{\ell_1}/N_{\ell_1}$.*

PROOF. Since there are no losses, $\forall_i \gamma_i = r_i$ and $\sum_{i\in F_{\ell_1}} r_i \leq C_{\ell_1}$.

Assume for contradiction, $\exists i \in F_{\ell_1}$ s.t. $r_i < C_{\ell_1}/N_{\ell_1}$.

Since ℓ_1 is the minimum bottleneck link, the fair rate f_ℓ of any other link ℓ as defined by Eq. 12 is greater than or equal to C_{ℓ_1}/N_{ℓ_1}. Set r_i' equal to C_{ℓ_1}/N_{ℓ_1} and the new output rate at each link will satisfy $\gamma_i' = r_i' > \gamma_i$. So this profile of rates cannot constitute a Nash Equilibrium.

Now assume WLOG, that flow $1 \in F_{\ell_1}$ becomes the leader in a Stackelberg Equilibrium. If $r_1 = C_{\ell_1}/N_{\ell_1}$, then using a similar argument as above, Nash Equilibrium of the remaining flows $i \in F_{\ell_1}, i \neq 1$ is when $r_i = C_{\ell_1}/N_{\ell_1}$. In this case γ_1 becomes equal to C_{ℓ_1}/N_{ℓ_1}. So any Stackelberg Equilibrium with flow 1 as leader should have $\gamma_1 \geq C_{\ell_1}/N_{\ell_1}$.

If $r_1 > C_{\ell_1}/N_{\ell_1}$ then the Nash rate of the remaining flows will be at-least C_{ℓ_1}/N_{ℓ_1}. Therefore flow 1 will have losses. Hence the proof. □

To extend Claim 1 to the subsequent bottleneck links, we use the flow removal property of Lemma 9.3. Since the flows in F_{ℓ_1} do not get any losses, they may be removed from the network without changing the fair rate f and the rate constant κ at any other link.

Therefore we have the following property:

In any Nash or Stackelberg Equilibrium without losses, any flow $i \in F_{\ell_k}$ s.t. $i \notin F_{\ell_{k'}}$ where $k' < k$ will have $\gamma_i = r_i = C_{\ell_k}^k/N_{\ell_k}^k = \mathcal{M}_i$. This is the set of max min fair rates and will hence constitute the unique Nash and Stackelberg Equilibrium without losses.

PROOF. Theorem 4.4: Follows directly from Lemma 4.5. □

LEMMA 9.5. *Let U_i be the utility function for user i. If $\forall_i U_i \in U^{\gamma l}$, then in any Nash or Stackelberg Equilibrium, no user will experience losses.*

PROOF. [**Nash Equilibrium**]: Suppose that for user i, $\gamma_i < r_i$. So the loss-rate is $l_i = r_i - \gamma_i > 0$. If user i changes its input rate to $r'_i = \gamma_i$, its input rate at the first link is decreased. Using Lemma 9.2 if the output rate should either stay the same, or increase, or become equal to the new input rate. Since $r'_i = \gamma_i$, the output rate at this link becomes equal to its new input rate, γ_i. The same argument applies to all the following links in the path. Therefore $\gamma'_i = \gamma_i$. The new loss rate is therefore zero. Since loss rate is decreased and output rate is same as before, user i's utility increases. So in any Nash Equilibrium, no user can have losses.
[**Stackelberg Equilibrium**]: In any Stackelberg Equilibrium, the leader cannot have any losses, else it can increase its utility by decreasing the input rate to its current output rate. All the other flows are in Nash Equilibrium, so by a similar argument they also cannot have any losses. □

PROOF. Theorem 4.5: From Lemma 9.5, any Nash or Stackelberg Equilibrium with utility functions $U_i \in U^{\gamma l}$ cannot have losses. If the loss rate is zero, then the a user's utility becomes only a function of its output rate and hence belongs to U^{γ}. Then using Lemma 4.5, it follows that the unique Nash and Stackelberg Equilibrium is at $\underline{\mathcal{M}}$. □