An Abstract Framework for Managing Non-Cooperative Flows in Congested Inter-Networks

K. Chandrayana and S. Kalyanaraman
ECSE Dept., Rensselaer Polytechnic Institute

Abstract— **In this paper we propose an analytical model for managing non-cooperative or mis-behaving flows in the Internet by manipulating congestion penalties conveyed to them. We show that a penalty transformation derived from the Duality formulation of Utility Maximization problem can map a user's utility function, U_s, to any objective utility function, U_{obj}. These penalty transformation modules can be completely implemented at the edge and require (Explicit Congestion Notification) ECN support. We have analyzed the framework and evaluated it for both single and multi bottleneck scenarios. This framework can also be used for providing more robust usage and flat rate pricing schemes and also for disassociating the fairness from user's rate control schemes.**

I. Introduction

Recently congestion control schemes have been evaluated and proposed using optimization frameworks [5], [6], [7]. In these papers, the resource allocation problem is proposed as 1) individual users maximizing their utility functions and 2) network maximizing every user's utility function given the network capacity constraints.

A significant result of these works is that they show the existence of stable rate adaptation schemes. The analysis also shows that the equilibrium rate allocation is very closely tied with the utility function the user chooses to maximize. *This association of equilibrium rate allocation with the utility function might prompt sources to choose a utility function (and hence an aggressive congestion control scheme) which yields them higher rate allocations than other competing sources*. Such a choice of utility function will still optimize the network and keep it stable, though at the cost of unfair allocations amongst users.

Further the Active Queue Management (AQM) schemes try to manage the congestion and thus do not differentiate between flows. As such the rate allocations in the network can be unfair, if all the users choose to maximize different utility functions. Therefore, *Can the fairness criteria be decided by the network irrespective of the utility function users choose to maximize?* We call this problem of unfair rate allocation as *Non-Cooperative Congestion Control*.

Constant Bit Rate (CBR) sources also pose the problem of unfair allocations. However CBR sources do not react to congestion indications hence we would need some sort of active policing mechanism like packet dropping, shaping etc. Further quite a few sites block UDP connections, as such it is imperative to look out for other possible forms of selfish rate-control schemes. Hence in this work we concentrate on patrolling the mis-behavior of *selfish responsive flows*, i.e. flows which react to loss or congestion indications by cutting down their rates. These flows can be obtained by small modifications to TCP's rate control algorithm. Thus they can go through any network and their selfish behavior poses a problem in the network. Now we outline the non-cooperative congestion control problem.

Suppose the network assumes that all the users are maximizing the utility function, U_{obj} and derive their rate control from it. We call, all the users who are actually maximizing U_{obj} as *cooperative or conformant* users. On the other hand, users who choose to maximize a utility function, U_s such that $U_s \neq U_{obj}$ are called *non-cooperative* users. Specifically, we call all the users who have higher marginal utility than U_{obj} (i.e. $U_s^{'} \geq U_{obj}^{'}$) as non-cooperative. This is because, the network allocates more resources to users who have higher marginal utility.

In this paper we show how a utility function can be mapped to a family of congestion control schemes, and thus define non-cooperative realms. Then once these non-cooperative sources have been identified, we show through analysis that by transforming their penalty function we can make these sources to behave as if they are conformant sources, i.e. map U_s to U_{obj}. In summary, we propose a framework for policing non-cooperative flows in the Internet by conveying a higher penalty to them. *Specifically, we propose a modification to the dual formulation [7] to map a user's utility function, U_s, to the network's objective utility function, U_{obj}*. Further, these penalty transformation agents *can be placed on the network edges* and we can choose to re-mark either the packets or acks. (Function similar to re-marking the ack stream is already being done [9]. Packeteer boxes *pace* the ack stream and are widely

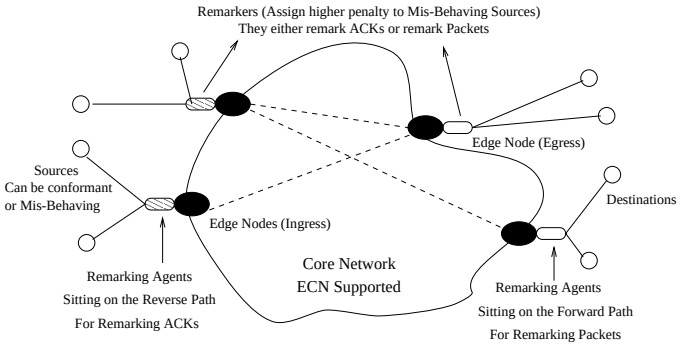


Fig. 1. Model for policing Non-Cooperative users through Penalty Transformation.

deployed over the Internet.) Figure 1 shows the model for policing non-cooperative users.

Through this utility function transformation we can disassociate the fairness criteria from the user's rate control scheme. So now the network provider can choose the fairness criteria he wants to deploy on his network, find a utility function which achieves it and then map every flow to that utility function. Also, this could lead to devising schemes wherein we can restrain the more aggressive flows and have some form of social optimum. This framework could also find application in pricing especially those of usage based or flat rate pricing. By having all users conform to a particular rate control scheme, the network ensures that it is fair to all users and hence makes usage based or flat rate billing more meaningful.

The model presented in this paper assumes that the utility functions are strictly concave and increasing or *elastic*. However, the model will also hold for strictly increasing pseudo-concave functions. Also re-marking packets (acks) is used as method of conveying penalties in this model, which might be extended to dropping packets. We haven't evaluated dropping as a means of conveying penalties. In cases where a flow uses a single exit (or entry) router the model is immune to path-asymmetry. However for all other cases path-asymmetry needs to be explored further. In spite of these limitations we believe this is a first step to modeling and managing non-cooperative flows.

Scheduling algorithms can also achieve the task of disassociating the fairness property from the user's rate control scheme. However, this choice would require placement of schedulers throughout the network. Clearly, this is not a readily deployable solution. Hence we need to look at schemes which can disassociate fairness from user's rate control scheme and which require minimal upgrades.

We implemented this framework in NS with the penalty function transformation agents placed in the reverse path to re-mark the ACKs. We evaluated it for various single and multi-bottleneck topologies. Out results show that the framework can "re-map" any non-cooperative user to co-operative user for any network scenario, if the utility function of the user is known to the network. Further, the framework is robust and works well even in the presence of background web-traffic and reverse-path congestion. However, when the utility functions of the user are not precisely known (or the utility function estimation error is more than 5%) the model cannot fully re-map the utility functions. In cases where we under-estimate the aggressiveness of non-cooperative flow the selfish user is not sufficiently marked and hence still grabs more bandwidth. However in spite of these estimation errors, these shares are still more fair than the case when there was no re-marking present.

The rest of the paper examines the policing of non-cooperative sources in detail. In Section II we first describe the system and discuss the dual formulation [7]. In Section III we present the non-cooperative congestion model. We present the simulation setup in Section IV, results in Section V and discussions on merits and drawbacks of the scheme in Section VI. Finally we present the conclusions and future work in Section VII.

II. Literature Survey

Recently, quite a few optimization models have been proposed for network flow optimization. In [5] the authors analyze the stability and fairness of network under primal and dual formulation. The dual formulation has also been discussed in [7] where gradient projection method is used to solve the problem. A penalty function approach to solving the network problem has been suggested by the authors in [6]. Also, the current TCP implementations have been mapped to optimized rate control algorithm in [8] [6].

In these optimization models a user *s*, is described with the help of it's rate, x_s, a utility function U_s and the set of links which he uses, *L(s)*. It is further assumed that the rates bounded i.e., $m_s \leq x_s \leq M_s$. It is assumed that the utility functions are *increasing* with rates and *strictly concave*. The network is identified with links *l* of capacity C_l. The set of users using a link, *l*, is given by *S(l)*.

The optimization problem can then be defined as user's trying to maximize their individual utility functions and the network trying to maximize the resource allocation subject to link capacity constraints. Thus the primal problem can be defined as:

$$maximize \sum_{s \in S} U_s(x_s) \tag{1}$$

$$subject\ to \sum_{s \in S(l)} x_s \leq C_l, \quad \forall l \tag{2}$$

for all $x_s \geq 0$. The dual formulation, D(p), for the above

problem was defined by Low in [7] as:

$$D(p) = \underbrace{min}_{p \geq 0} \sum_{s \in S} (U_s(x_s) - \sum_l p_l x_s) + \sum_l p_l C_l \quad (3)$$

The authors in [7] show that using the Karush Kuhn Tucker (KKT) conditions and gradient projection algorithm the dual yields the following update algorithm

$$x_s(t) = U_s^{'-1}(\sum_l p_l) \quad (4)$$

$$p_l(t+1) = [p_l(t) + \gamma(\sum_{s \in S(l)} x_s - C_l)]^+ \quad (5)$$

Since the primal is strictly concave and the constraints are linear, there is no duality gap and hence dual optimal is also primal optimal. Further the strict concavity entails an unique global optimum, (x_s^*, p^{s*}) where $p^s = \sum_l p_l$. Also though the primal optimal, x_s^* is unique, we may not have a unique dual optimal p_l^* but instead we have a unique optimum end-to-end loss probability for every source, p^{s*}.

III. Non-Cooperative Congestion Control Framework

From the discussion in the previous section we can conclude that both the rate control algorithm and the equilibrium rate can be associated with the utility function user chooses to maximize (equation (4,5)). However, given the same price being communicated by the network, the equilibrium rates can be unequal. Thus even though the network doesn't desire to be perceived unfair, a bias in equilibrium rates can be created by choosing two different utility functions. We illustrate this through an example from [6].

Consider a bottleneck link where two set of rate control schemes compete for the bandwidth. The utility function for set 1 is given by $U_s(x_s) = w_s log(x_s)$ and that for set 2 is given by $U_s(x_s) = -w_s x_s^{-1}$, where w_s represents the weight assigned to the flow. Let there be 50 sources each in Set 1 and Set 2. Assume that the link capacity to be 300, weights to be 1, the round-trip time (RTT) for all sources to be same. Then the throughput seen by each source can be obtained by solving the following optimization problem:

$$max \quad \sum_{i=1}^{50} log x_i - \sum_{j=51}^{100} \frac{1}{x_j} \quad (6)$$

$$subject\ to \quad \sum_{i=1}^{50} x_i + \sum_{j=51}^{100} x_j \leq 300 \quad (7)$$

and $x_i, x_j \geq 0\ \forall i, j$. Solving this problem yields $x_i = 4.0, i \in \{1, ..., 50\}$ and $x_j = 2.0, j \in \{51, ..., 100\}$. Thus even though the network is fair, the equilibrium rate depends on the rate control algorithm chosen by the sources.

Thus the sources can choose rate control schemes which yield higher rate allocations. Another important point to note is that even though the sources are cooperative (i.e. they react to congestion indication) they still can be unfair. Therefore it is desirable to move the fairness criteria away from the user's rate control scheme to the network. That way the network not only has the flexibility of being fair, but more importantly it can choose the fairness criteria it wants to provide. Now we shall describe the non-cooperative congestion control framework.

Consider the network model as described in Section II where the user's are maximizing the utility function U_s. Assume, that the network decides that the final equilibrium rate allocation should be, as if every user chose to maximize a utility function of U_{obj}. The rate updation algorithm (and thus equilibrium rates) of the users is given by equation (4). Now, if we communicate a link price $f(p_l)$, instead of p_l, then the user-rate updation algorithm will be

$$x_s(t) = U_s^{'-1}(\sum_{l \in L(s)} f(p_l))$$

Further, if we choose $f(p_l) : f(p_l) \geq 0, \forall p_l$, *f(0) = 0* and the following condition holds true

$$\sum_{l \in L(s)} f(p_l) = U_s^{'}(U_{obj}^{'-1}(\sum_{l \in L(s)} p_l)) = g(\sum_{l \in L(s)} p_l) \quad (8)$$

then the rate updation algorithm algorithm becomes

$$x_s = U_s^{'-1}(\sum_{l \in L(s)} f(p_l)) \quad (9)$$

$$= U_s^{'-1}[U_s^{'}(U_{obj}^{'-1}(p^s))] \quad (10)$$

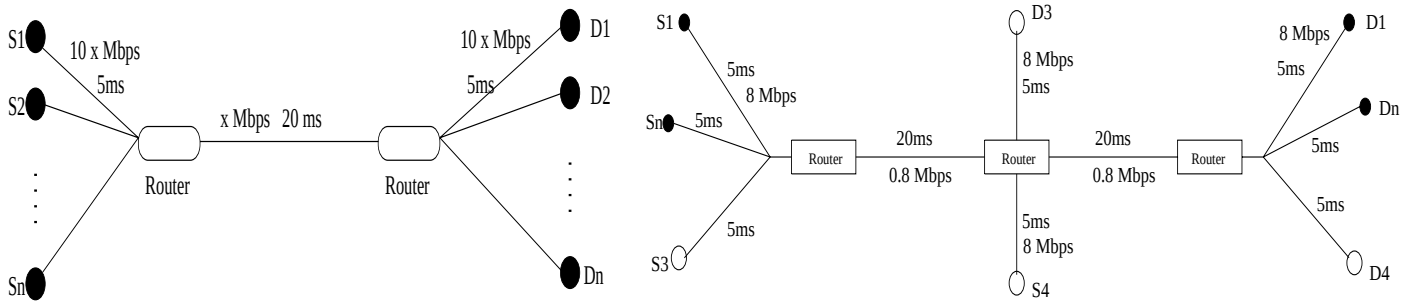
$$= U_{obj}^{'-1}(p^s) \quad (11)$$

where $p^s = \sum_{l \in L(s)} p_l$. From the above equation it is easy to see that by communicating a different price we have transformed the user's utility function from $U_s(x)$ to $U_{obj}(x)$. This transformation can be explained by the following modified dual:

$$D(p) = \sum_{s \in S} U_s(x_s) - \sum_l f(p_l)(\sum_{s \in S(l)} x_s - C_l) \quad (12)$$

where $f(p_l)$ is defined by equation 8. Now if, $f(p_l)$ is an increasing function in p_l and is always greater than 0; then this function will not change the minima (because $f(p_l)$ satisfies all the properties of Lagrangian multipliers).

Proposition 1: Given the non-negativity constraint on x_s and p_l and strictly concave utility functions U_s and U_{obj},



(a) Single Bottleneck Topology (b) Multi-Bottleneck Topology

Fig. 2. Topologies used in the Simulations.

the function $g(\sum_{l \in S(l)} p_l)$, $f(p_l)$ as defined in (8) are non-negative and strictly increasing in their argument.

Proof: See Appendix VIII-A

Using the KKT conditions and the gradient projection method we get the following rate and price updation rules

$$x_s = U_s^{'-1}(\sum_l f(p_l)) \tag{13}$$

$$p_l(t+1) = [p_l(t) + \gamma \frac{\partial}{\partial p_l} f(p_l)(\sum_{s \in S(l)} x_s - C_l)]^+ \tag{14}$$

The above formulation is however impractical to implement because it requires per-flow queuing and that too inside the network. Since upgrades in the network are hard to achieve consider the following update rule.

$$p_l(t+1) = [p_l(t) + \gamma(\sum_{s \in S(l)} x_s - C_l)]^+ \tag{15}$$

$$x_s(t+1) = U_s^{'-1}(\sum_l f(p_l(t))) \tag{16}$$

Proposition 2: Given the non-negativity constraint on x_s and p_l and strictly concave utility functions U_s and U_{obj}, the new update algorithm as defined in equation (15,16) still converges to the optimal point.

Proof: See Appendix VIII-B

Thus this update rule does not change the network objective, but still minimizes the dual function and converges asymptotically. The above update rule also does not change the core network, as we retain the price update rule as proposed in [7]. Further, the price being communicated to the user can be updated at the edge. We now state the algorithm for the edge re-marker as

Edge Marker's Algorithm:

- For each source, receive from the network the total price for the source's traffic as $p^s(t) = \sum_{l \in S(l)} p_l(t)$.
- Recalculate (or Re-mark) the new price for the source as

$$p^s_{new} = g(\sum_{l \in S(l)} p_l(t)).$$

- Communicate this *re-marked* price to the source.

The update algorithm for the network and the source are given by equation (15) and (16) respectively.

Theorem 1: Assume that utility functions, U_s, are increasing, strictly concave and continuously differentiable, and their curvature is bounded away from 0. Then starting from any initial rates in the interior of *X* and prices *p(0)* $\geq$ *0*, every accumulation point (x^*, p^*) of the sequence (*x(t), p(t)*) generated by the above algorithm and equations (15,16) is primal dual optimal.

Proof: See Appendix VIII-E.

IV. Implementation

We implemented the edge based re-marker in the NS (Network Simulator). The edge based re-marker was placed on the reverse path (i.e. on the reverse access link of the user) and re-marked the ACKs. The edge re-marker also estimated the loss rate for each flow and subsequently used it to re-mark the ACKs. For the purposes of estimating losses, we used Exponential Weighted Moving Average (EWMA) and the Weighted Average Loss Indication (WALI) methods of Equation-based rate control algorithm [4]. We updated these loss indications every RTT and we have assumed that the network knows the RTT of the flows. We also assumed that we know the utility functions of all the flows. In this paper we present the results for EWMA based loss-estimator. Similar results were obtained with WALI based estimator. For EWMA based system we gave 60% weight to the history, while with the WALI based estimator we measured samples over 8 windows to estimate losses. A more detailed discussion on the merits and demerits of these schemes can be found in [4].

For our simulation we used the congestion control and loss recovery mechanisms of TCP New Reno. Also in this paper, we disabled the delayed acknowledgments option. For simulating non-cooperative flows we used the Binomial Congestion Control scheme (BCCS) proposed in [1]. The window increase for BCCS is defined as

$$W_{t+R} \leftarrow W_t + \alpha / W_t^k \quad if\ no\ loss \tag{17}$$

$$W_{t+\delta t} \leftarrow W_t - \beta W_t^l \quad if\ loss \tag{18}$$

where α, β, k, l define the Binomial Algorithm. In this paper we fixed the values of α, β as 1 and 0.5 respectively. The throughput of Binomial Algorithm can be calculated to be as $x \propto \frac{1}{p^{\frac{1}{k+l+1}}}$ where *x* represents the throughput and *p* represents the loss or mark rate. But from equation (4), the marginal utility for a binomial scheme can be approximately calculated as $U_s^{'}(x_s) \propto p \propto \frac{1}{x^{k+l+1}}$ Also, it follows from the above discussion that any utility function which is defined as $U(x) \propto \frac{-1}{x^n}$ can be mapped to a family of binomial schemes such that *k+l=n*. This provides a way to map quite a few utility functions to a window based congestion control algorithm.

Therefore we use binomial schemes to generate non-cooperative traffic sources. Further in this paper *we have defined the flows which are TCP Friendly* [3], *i.e.* k+l = 1 *as cooperative while the flows which beat TCP Friendly flows are called non-cooperative*. From the above discussion it follows that *non-cooperative flows can be generated by choosing* $k + l < 1$. Henceforth, in this paper, we will use the *k* and *l* values to identify non-cooperative flows. However we need to point out that TCP Friendliness is just one special case of cooperation in this framework as the network may choose any other model of cooperation, i.e., any other choice of the U_{obj}.

Figure 2(a) shows the single bottleneck topology used in the simulations. The access links were configured at a rate 10 times greater than that of the bottleneck link. All the links use Random Early Drop (RED) queues with min thresh and max thresh set as buffer/3 and 0.8*buffer respectively, where buffer is the total bottleneck buffer length. Further, the weight was set as 0.002 and the marking probability for RED was set to 0.1. The RTT was 60ms and the packet size 500B.

Figure 2(b) shows a multi-bottleneck topology used in the simulation. The bottleneck buffer was set to 25 packets. We also evaluated our framework for another multi-bottleneck setup of bottleneck link of 10 Mbps, access link of 100 Mbps and a buffer of 250 packets. The link delays were kept the same. RED minimum and maximum threshold settings were similar to those of single bottleneck. Also for all the simulation setups (single or multi-bottleneck) the access link rate are always 10 times greater than that of the bottleneck link.

The maximum advertised window is set sufficiently high so that it does not constrain the actual window. The simulation time for each setup was 1500 seconds. We plot the throughput of competing flows in packets/sec, averaged over 20 round-trip times. We assumed that all the flows have infinite data to transfer.

V. Simulation Results

In the following section we present the results of the simulation for single bottleneck and multiple bottleneck topologies. We evaluate both these setups for different bottleneck link capacity and different type of non-cooperative flows. We also validate the robustness and correctness of the scheme with background and cross traffic. We first present the results for single bottleneck topology and follow it with other results in subsequent sections.

A. Single Bottleneck

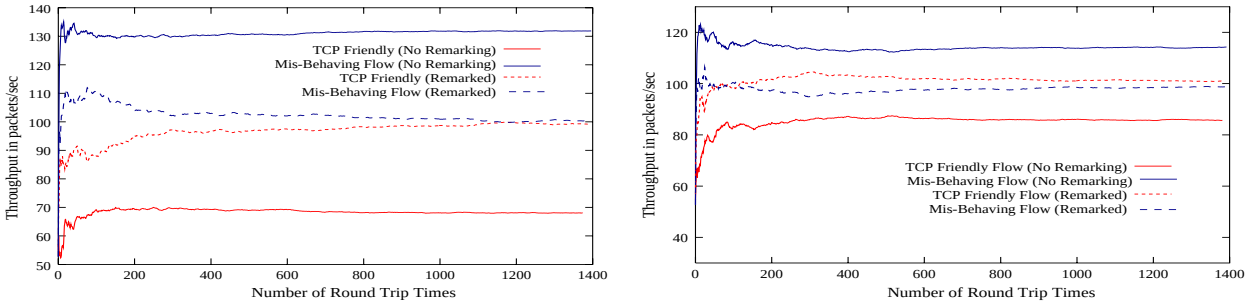
In figure 3 a) we present the throughputs of two competing flows on a single bottleneck of 0.8 Mbps with a buffer of 25 packets. Here, one of the flows is TCP, while the other is non-cooperative and is defined by $k = 0$ and $l = 0.5$. As the figure 3 a) shows, when we do not re-mark the non-cooperative flow, it garners more bandwidth than the TCP friendly flow. However, re-marking the non-cooperative flow makes the two flows to share the bandwidth equitably. Figure 3 b) show similar results where the non-cooperative user is defined by $k = 0, l = 0.8$.

Figure 4 shows the results for a set of 10 competing flows on a 10Mbps bottleneck and 150 packet buffer. The flow set comprises of 7 TCP Friendly flows while the remaining 3 flows are non-cooperative and are defined by $k = 0$ and $l = 0.5$. The bandwidth is shared equitably in presence of re-marking, however in absence of it misbehaving flows easily beat the TCP Friendly flows.

We also evaluated our scheme when every flow has a different utility function. Figure 5 shows the result for one such setup for a bottleneck bandwidth of 0.8 Mbps. In this simulation setup we have three flows, one TCP-Friendly flow and the others are defined as (k=0, l=0.5) and (k=0, l=0.2). We can see from the figure 5 that in the absence of re-marking, non-cooperative flows beat the TCP friendly flow; however when we re-mark the non-cooperative flows the bandwidth is shared fairly.

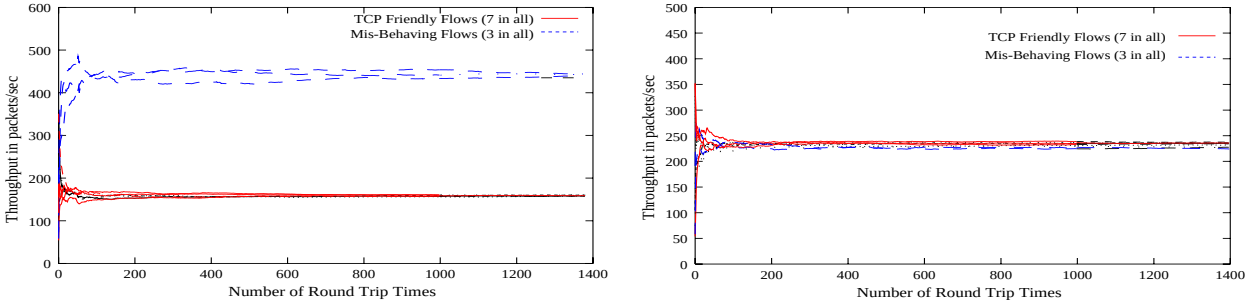
B. Multi Bottleneck Topology

In this section we present the results for multi-bottleneck topology (figure 2 b)). We define long flow as a flow which traverses both the bottleneck, whereas the short flows are defined as flows traversing only one bottleneck. In this simulation setup (0.8Mbps, 25 packet buffer), we first measured the optimal rate allocations when all the flows (long and short) are TCP friendly and plot them in 6 a). As expected, the short flows grab more share of the bottleneck because they have smaller RTTs and go through a single bottleneck as compared to the long flow. We then changed the short flows to be non-cooperative (k=0, l=0.5)



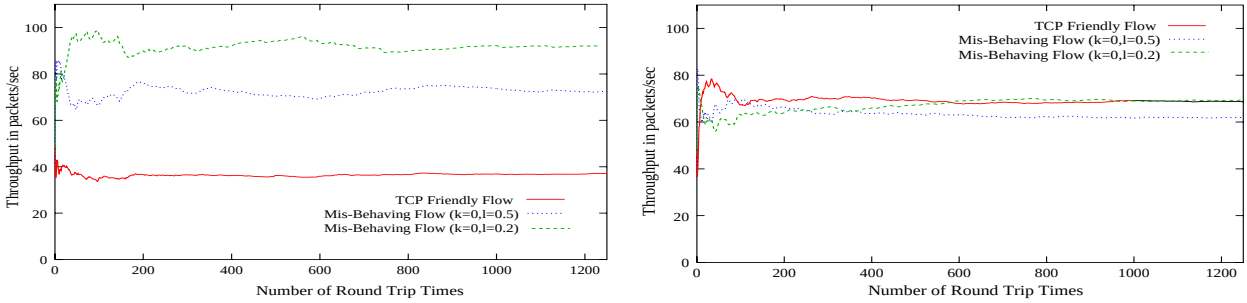
(a) k=0, l=0.5 (b) k=0, l=0.8

Fig. 3. **Single Bottleneck**: Throughputs (in pkts/sec) for two competing flows, one is TCP Friendly while the other is Non-Cooperative with and without Re-Marking.



(a) No Re-Marking (b) Re-Marking the Non-Cooperative Flow

Fig. 4. **Single Bottleneck**: Throughputs (in pkts/sec) for ten competing flows, where seven flows are TCP Friendly while three are Non-Cooperative (k=0, l=0.5) with and without Re-Marking.

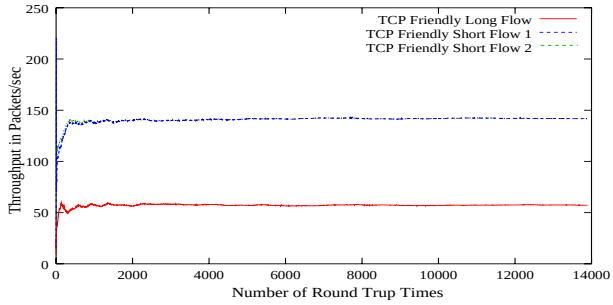


(a) No Re-Marking (b) Re-Marking the Non-Cooperative Flows

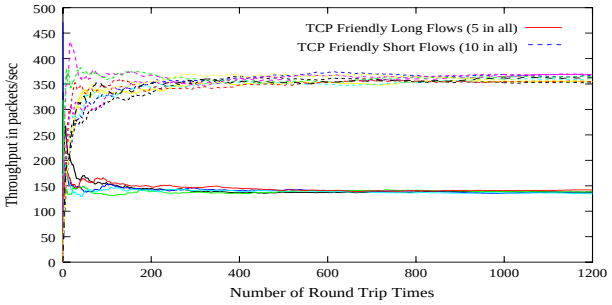
Fig. 5. **Single Bottleneck**: Throughputs (in pkts/sec) for three competing flows, where one flow is TCP Friendly while the other two are Non-Cooperative with (k=0, l=0.5) and (k=0, l=0.2) resp., with and without Re-Marking.

and plot the result in 6 b). The effect of mis-behavior is more pronounced in this case as the non-cooperative flows are trying to shut out the TCP friendly flow. However, when we used our model to re-mark the non-cooperative flows we see that (figure 6 c)) the flows now share the bandwidth fairly. More importantly, we see that the result in figure 6 c) is very similar to 6 a), i.e., *we have successfully mapped the utility function of the non-cooperative flows*. We also simulated the scenario where the long flows were non-cooperative and short flows TCP-Friendly. However due to space constraints we do not show those results here. There too our model was successful in mapping the utility function of non-cooperative flows.

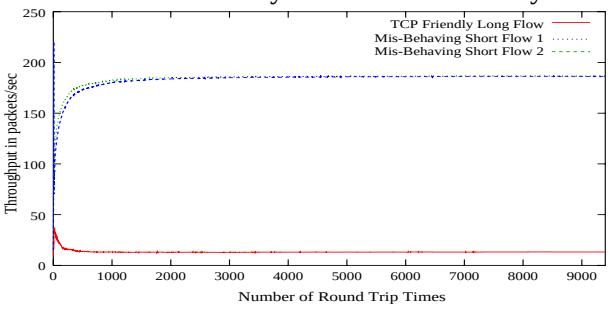
In figures 6 d), e) and f) we plot the results for a multi-bottleneck topology (10Mbps, 250 packets buffer) where on each bottleneck there are 5 TCP Friendly flows and 5 non-cooperative flows (k=0, l=0.5). Figure 6 (d) plots the throughput of long and short flows, if they all were TCP Friendly. As expected the longer flows get a smaller share of the bottleneck than the shorter flows. In Figure 6 (e), we changed the shorter flows to act as non-cooperative flows and plot the throughput, and it can be seen that the non-cooperative shorter flows conveniently beat down the TCP friendly flows. However, in presence of re-marking, (Figure 6 (f)) the non-cooperative flows are conveyed higher price by the edge-re-marker and hence now share the bottleneck more favorably with the longer flows. Once again, we see that *re-marking tends to achieve the same performance as those as if all the flows were TCP Friendly*.



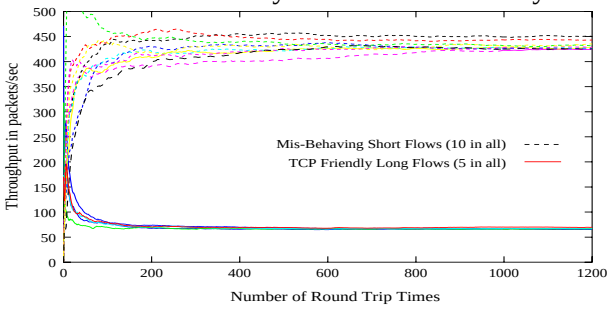
(a) 2 Competing Flows on each bottleneck, 1 Long and 1 short: Every Flow is TCP Friendly



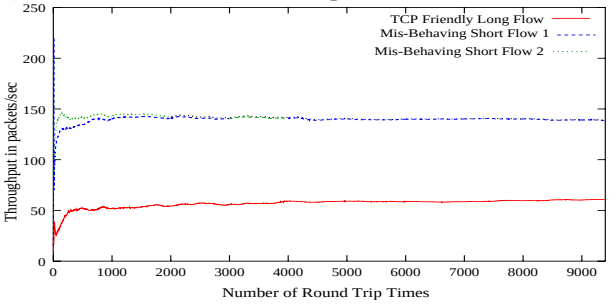
(d) 10 Competing Flows on each bottleneck, 5 Long and 5 short: Every Flow is TCP Friendly



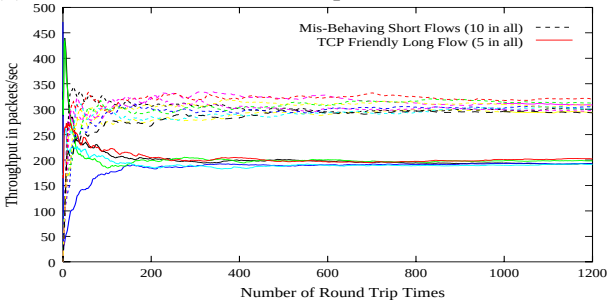
(b) Short Flows are Non-Cooperative, No Re-Marking



(e) Short Flows are Non-Cooperative, No Re-Marking

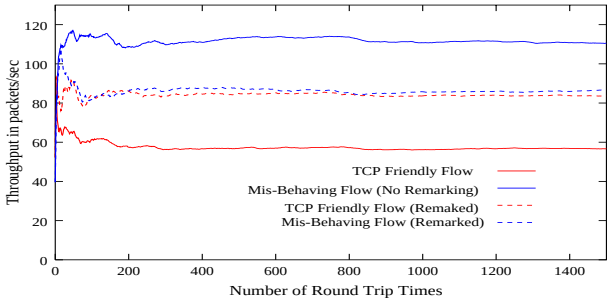


(c) Short Flows are Non-Cooperative, Re-Marking

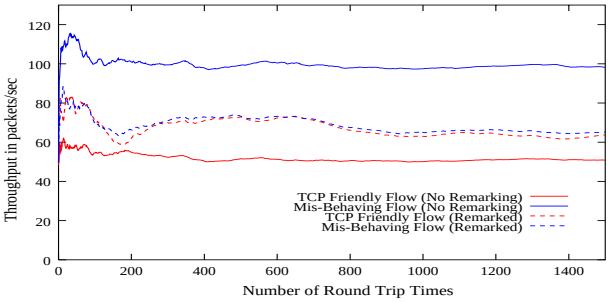


(f) Short Flows are Non-Cooperative, Re-Marking

Fig. 6. **Multi Bottleneck**: Throughputs (in pkts/sec) for competing flows (2 and 10), where the long flows are TCP Friendly while the short flows are Non-Cooperative with (k=0, l=0.5). Ideal bottleneck shares for the long and short flows for 2,10 competing flows are plotted in figures (a) and (d) respectively.



(a) 15% noise



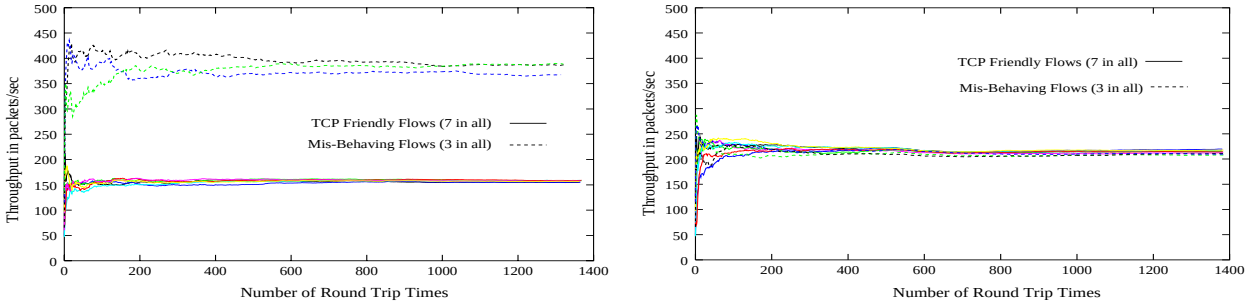
(b) 25% noise

Fig. 7. **Background Traffic**: Throughputs (in pkts/sec) for two competing flows in a single bottleneck topology, where one flow is TCP Friendly while the other is Non-Cooperative with (k=0, l=0.5). Also there is web-traffic in the background.

C. Background Traffic

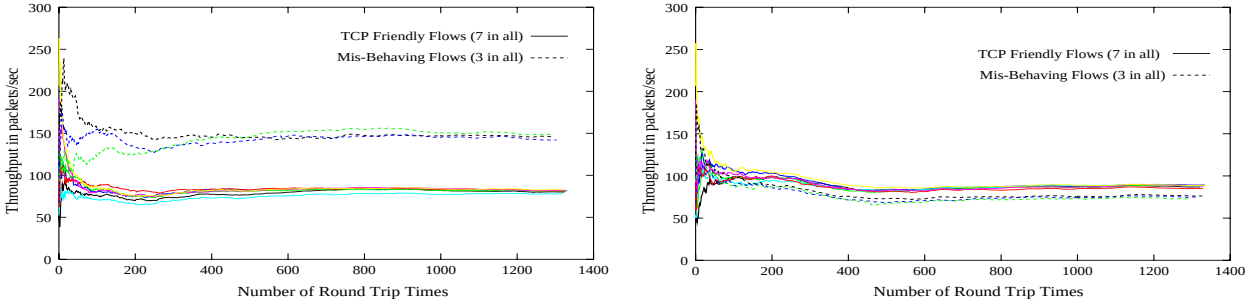
In this section we evaluate the framework in presence of noise-like mice traffic. HTTP sources were added to the persistent non-cooperative and conformant sources. Each http page sends a single packet request to the destination, which then replies with a file of size which was exponentially distributed with 12 1Kb packets. After a source completes this transfer it waits for a random time, which was exponentially distributed with a mean of 1 second and then repeats the process.

Two sets of simulations were conducted for the single



(a) No Re-Marking (b) Re-Marking

Fig. 8. **Background Traffic**: Throughputs (in pkts/sec) for 10 competing flows in a single bottleneck topology, where 7 flows are TCP Friendly while the other 3 are Non-Cooperative with (k=0, l=0.5) **with 20% noise**.



(a) No Re-Marking (b) Re-Marking

Fig. 9. **Background Traffic**:Throughputs (in pkts/sec) for 10 competing flows in a single bottleneck topology, where 7 flows are TCP Friendly while the other 3 are Non-Cooperative with (k=0, l=0.5) **with 65% noise**.

bottleneck case. In the first simulation, there were two persistent flows (one non-cooperative and the other TCP Friendly) competing for a bottleneck of 0.8 Mbps. 2 and 4 http sources were added to generate 15% and 25% noise (ie the http sources occupied 15% and 25% of the bottleneck bandwidth). The results for this simulation are plotted in figure 7. Again, it can be seen that the re-marking works well in the presence of noise and the bottleneck is shared equitably. In another simulation we increased the number of competing persistent flows to 10 and of these, 7 flows were TCP Friendly while the remaining 3 where non-cooperative (k=0,l=0.5). The bottleneck bandwidth for this simulation was 10Mbps and a buffer of size 150 packets. Also in this setup we increased the noise sufficiently high to validate the robustness of the scheme in presence of many flows and noise. Figures 8 and 9 plot the results for the cases where the noise traffic is 20% (25 http sources) and 65% (80 http sources) respectively. Figure 9 shows the robustness of the scheme when sufficiently high (65%) noise is present in the network and the re-marker still manages to efficiently patrol non-cooperative users.

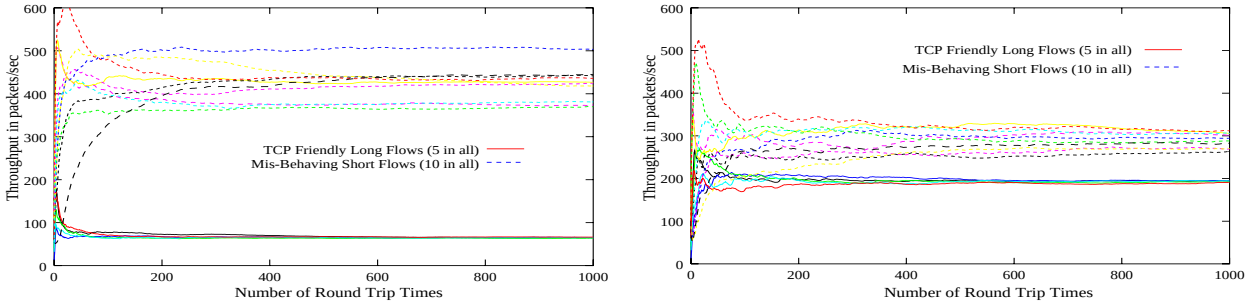
D. Cross Traffic

In this section we present the results for our penalty function transformer when two way traffic is present. We evaluate this scenario with the multi-bottleneck topology, where we have 5 TCP Friendly long flows and 5 non-cooperative (k=0, l=0.5) short flows on each bottleneck. Additionally, on the reverse path, there are 5 TCP Reno flows on each bottleneck. The bottleneck bandwidth for this simulation was 10Mbps and a buffer of size 250 packets. Re-Marking, once again achieves equitable sharing of the bottleneck (as shown in Figures 10 (a) and (b)).

E. Effect of Inaccurate RTT Estimate

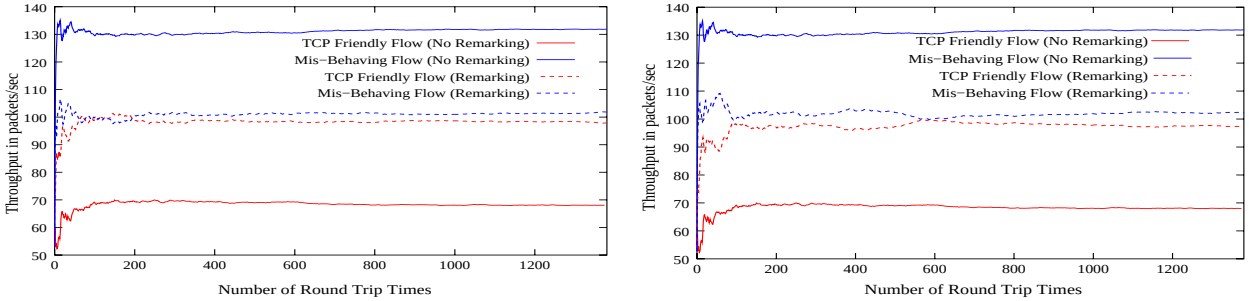
In this section we investigate the effect of inaccurate RTT estimates. In all our previous simulations we assumed that the network knows the RTT of the flows. We used these RTT estimates to update our congestion indication estimations. For the results presented in this section we looked at two cases, one when we under-estimated the RTT and the other when we over-estimated it. We present the results with a single-bottleneck of 0.8Mbps, 25 packet buffer and 2 competing flows.

Figure 11 a) shows the results when the RTT was under-estimated as 0.05 (instead of 0.06). Figure 11 b) shows similar results when we over-estimated the RTT as 0.07. The figures suggest that inaccuracy in RTT estimates alters the convergence speed to the optimal point; a larger value of RTT will slow down the convergence while a smaller value will increase the convergence. However, from both the results its easy to see that the effect of inaccurate RTT estimation is not pronounced and the model works well. We ran simulations with higher degree of multiplexing and



(a) No Re-Marking (b) With Re-Marking

Fig. 10. **Cross Traffic**: Throughputs (in pkts/sec) for 10 competing flows in a multi-bottleneck topology, where on each bottleneck there are 5 TCP Friendly flows and 5 Non-Cooperative with (k=0, l=0.5), with two-way traffic.



(a) Under-estimation of RTT (rtt=0.05s) (b) Over-Estimation of RTT (rtt=0.07s)

Fig. 11. **Inaccurate RTT Estimates**: Throughputs (in pkts/sec) for 2 competing flows in a single-bottleneck topology, where one flow is TCP Friendly flow while other is Non-Cooperative with (k=0, l=0.5), when network has inaccurate RTT estimates.

came to a similar conclusion. Due to reasons of space constraints we do not present those results here.

F. Effect of Inaccurate Utility Function Estimate

Till now we have assumed that the network knows the utility function of the flows. Since utility functions are not being explicit conveyed to the network therefore we will need to estimate them. Thus we need to explore the effect of inaccurate utility function estimates. In this section we evaluate the model's sensitivity to utility functions; when the utility functions are under-estimated and second when they are over-estimated. Under-estimation here refers to the case when we estimate the utility function to be less aggressive than it really is, i.e. when $k + l$ values are reported to be larger than the actual values. Over-estimation refers to the case where we report the flow to be more aggressive than it really is, i.e. $k + l$ values are reported to be smaller than the actual values. We present the results with a single-bottleneck topology (figure 2 a)) for 2 flows.

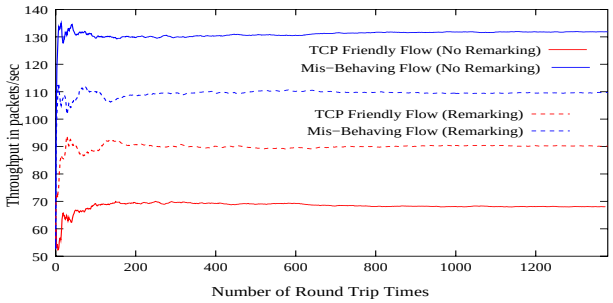
Figure 12 a) shows the results when the utility function was under-estimated as 0.6 (instead of 0.5). Figure 12 b) shows similar results when we over-estimated it as 0.4. It can be seen from the results that the model is sensitive to inaccurate estimate of utility functions. When we under-estimated the utility function ($k+l = 0.6$) the model didn't penalize the mis-behaving flow much, and as such it still garners more bandwidth than the TCP flow. In the case of over-estimation ($k + l = 0.4$) we see that the network penalizes the mis-behaving flow more and consequently brings it share down below the TCP Friendly flow.

However, the estimation errors pointed out in the simulation are large (the error is 20% since we estimate the $k + l$ values as 0.5 ± 0.1). We evaluated the model for two other error estimates, 10% and 5% and report the result for the 5% error case in figure 13. As expected, as the estimation error decreases the model starts to get better. Further we found that for estimation errors of more than 5% the model does not penalize (or over penalizes) the mis-behaving flow much and it consequently has a larger (or smaller) share at the bottleneck. However in spite of these estimation errors, these shares are still more fair than the case when there was no re-marking present. For estimation errors of less than or equal to 5% the model worked well (figure 13). We evaluated the model for different simulation setting where we had 10 flows (5 mis-behaving, 5 friendly) and came to a similar conclusion.

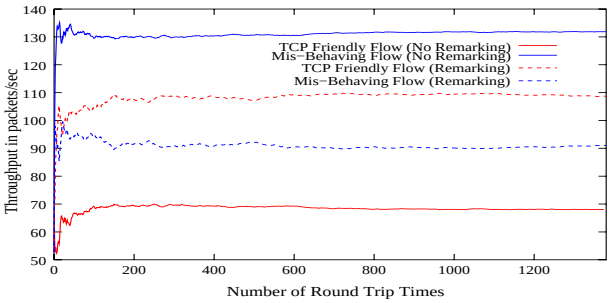
VI. Discussion

In this paper we have proposed an abstract model for modeling and managing non-cooperative flows. We believe this is the first work in this direction and an area which needs to be explored further. In this section we will debate the merits and the limitations of the model.

In this paper we have modeled the utility function of

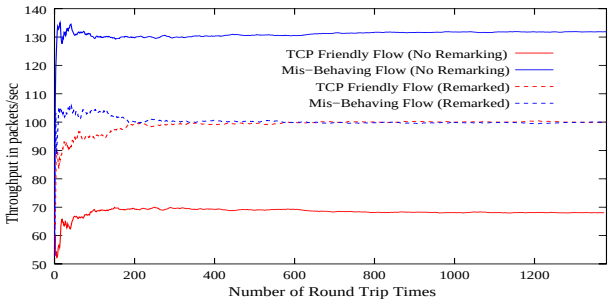


(a) Under-estimation of Utility Function: In-sufficient Re-marking, Misbehaving Flow Still Wins

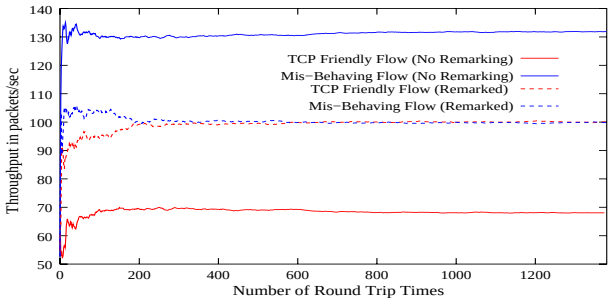


(b) Over-Estimation of Utility Function: Excessive Re-marking, Misbehaving Over Penalized (Loses to TCP)

Fig. 12. **Inaccurate Utility Function Estimates, 20% Estimation Errors**: Throughputs for 2 competing flows in a single-bottleneck topology, where one flow is TCP Friendly flow while other is Non-Cooperative with (k=0, l=0.5), when network has inaccurate estimates of source's utility function.



(a) Under-estimation of Utility Function: Bottleneck is still shared equitably because estimation errors are small



(b) Over-Estimation of Utility Function: Bottleneck is still shared fairly because estimation errors are small.

Fig. 13. **Inaccurate Utility Function Estimates, 5% Estimation Errors**: Throughputs (in pkts/sec) for 2 competing flows in a single-bottleneck topology, where one flow is TCP Friendly flow while other is Non-Cooperative with (k=0, l=0.5), when network has inaccurate estimates of source's utility function.

non-cooperative flows as strictly increasing and concave. Clearly, this is just one part of the non-cooperative flows, for CBR flows, pseudo-concave utility etc would complete the non-cooperative flow realm. CBR sources could be modeled as saturation functions (as used in control systems). However these sources do not react to marks and as such we would need to look at dropping as the means of penalty. Since dropping is an irreversible function, the penalty function transformers should be placed on the forward path, at the egress nodes. But then path-asymmetry issues can complicate the placement of the penalty function transformers. We leave the policing of CBR sources as future work. Real time application needs are often modeled using pseudo-concave functions. The model proposed in this paper, would still be valid for inelastic utility function provided that they are strictly increasing in their argument and pseudo concave.

We have assumed in this paper that the utility function of the user is known. Though not a trivial problem, this is certainly achievable. Given the knowledge of RTT of a flow, sampling window growth and then using Maximum Likelihood Estimators can help track the window scaling parameters, k, l. Recently, studies have been made in this direction, though it still lacks a comprehensive analytical and experimental solution. In [10] the authors have evaluated different TCP flavors of web-servers, chiefly TCP Tahoe, Reno, New Reno and Sack. This tool's, TBIT (TCP Behavior Inference Tool [10]) architecture can be leveraged to estimate the scaling parameters of TCP like rate control schemes. The scheme proposed in this paper is sensitive to loss-estimation. Techniques for loss estimation have been discussed in detail in [4]. However, these techniques are still empirical at best and need to to be evaluated further.

Path asymmetry is another issue which we need to look in detail. If a single exit router is used by the flow then the model is immune to path-asymmetry problems in the network. In this case, the penalty function transformers can be placed at that exit router and it can re-mark the flow. The same observation holds true if there exists a single ingress router for a flow. Both unique entry or exit routers is generally true in the present Internet. However, if different exits and entry routers are used for any flow then we need to study the effect of path-asymmetry.

Despite these limitations, we believe our work is a first step in modeling and managing non-cooperative flows. The incentives for such an approach are clearly high. We

have showed in this paper that we can effectively disassociate the fairness criteria from the user's rate control algorithm and move it to the network. The solution proposed in this paper can be implemented at the network edges and requires ECN support. Further we have a flexibility of choosing either to re-mark the packets or acks. In cases where we do not have access to the packet-stream we can re-mark the ack-stream and achieve the goals of the model. (Packeteer boxes, deployed widely on the Internet, already do a similar work by accessing the ack-stream and pacing the acks [9]). For cases where we have access to the packet stream the model could be extended to use dropping as a means of communicating penalties. In such a scenario this model could work without ECN support or i.e. with a network of Drop Tail queues only.

This framework can also provide more robust and fair usage based and flat rate pricing strategies. In the absence of a framework that protects sources against non-cooperative users, both these schemes fall short. If a non-cooperative user is priced for connection duration, he gets to use the same resource at a much cheaper price. Similarly, charging based on traffic volume does not take care of fairness in the network. However, with this model we can protect flows from selfish users. The reader is referred to [2] for a more detailed discussion on the achievable pricing strategies within this framework.

VII. Conclusions and Future Work

In this paper we have proposed an abstract model for modeling and managing non-cooperative flows. Specifically we propose a framework to map a user's utility function, U_s, to any objective utility function, U_{obj}, by manipulating congestion penalties. These penalty transformation agents can be completely implemented at network edges and require ECN support from the network. This framework can be used for managing selfish flows, providing more robust usage and flat rate pricing schemes and disassociating fairness from user's rate control schemes.

We have analyzed the framework and evaluated it for various single and multi-bottleneck scenarios. In the cases where the network has precise estimate of source's utility function (or the estimation error is less than 5%) the simulation results match the analysis and the model re-maps the mis-behaving source's utility function fully. Then the mis-behaving and the compliant flows share the bottleneck fairly. Further the model is robust and works well even in presence of high background (web) traffic and reverse path congestion. Also inaccurate RTT estimates do not have a very profound effect on the model's correctness. However when the utility function estimation errors were more than 5% the model does not fully correct the mis-behaving flow(s), but still considerably improves the fairness at the bottleneck (as compared to the scenario when there was no remarking).

We are currently working on ways to estimate the utility function of the sources by sampling either the packet-stream or ack-stream. Also we are working to extend the model to use dropping as a means of communicating congestion penalties.

VIII. Appendix

The assumptions used in the paper are as follows

- *A1:* The Utility functions are continuous, strictly concave and increasing in their arguments. Further the rates are bounded by *I: [m_s, M_s]*.
- *A2:* The curvature of U_s are bounded away from 0 on *I*, i.e. $-U_s^{''}(x_s) \geq 1/\alpha_s > 0$.

A. Proposition 1: *Under assumptions A1,A2 the function $g(\sum_{l \in S(l)} p_l)$, $f(p_l)$ as defined in equation (8) are non-negative and increasing in their argument.*

Proof: Define $p^s = \sum_{l \in S(l)} p_l$. Note $g(p^s) = U_s^{'}(U_{obj}^{'-1}(p^s))$. Recognizing that $U_{obj}^{'-1}(p^s)$ is just x_s from equation (4), we can rewrite $g(p^s)$ as $g(p^s) = U_s^{'}(x_s(p^s))$. Since $U_s(x_s)$ is increasing and strictly concave in its arguments hence $U_s^{'}(x_s) \geq 0$. Hence, $g_(p^s)$ is greater than 0. Let's define $F(p^s) = U_{obj}^{'}(p^s)$ and it's inverse as $H(p^s) = F^{-1}(p^s)$. Therefore, $H(F(p^s)) = p^s$.
Now differentiating both sides with respect to p^s we get,

$$H^{'}(F(p^s)) \cdot F^{'}(p^s) = 1 \quad (19)$$

$$or \qquad (U_{obj}^{'-1}())^{'} = \frac{1}{U_{obj}^{''}(p^s)}. \quad (20)$$

Now, differentiating $g(p^s)$ with respect to p^s we get

$$\begin{aligned} g^{'}(p^s) &= U_s^{''}(U_{obj}^{'-1}(p^s)).(U_{obj}^{'-1}(p^s))^{'} \\ &= U_s^{''}(\cdot)(U_{obj}^{'-1}(\cdot))^{'}. \end{aligned} \quad (21)$$

Since U_s and U_{obj} are strictly concave therefore $U_s^{''}(), U_{obj}^{''}() < 0$ and from equation (20) we conclude that $g^{'}(p^s)$ is greater than 0. Combining $g^{'}(p^s) > 0$ and the definition of $f(p_l)$ (equation 8) we conclude $f^{'}(p^l) > 0$.

B. Proposition 2: *Under assumptions (A1, A2), the new update algorithm as defined in equation (15,16) still converges to the optimal point.*

Using the equation 12 and differentiating it with respect to time we get

$$\frac{d}{dt}D(p) = \frac{d}{dp_l}\left(f(p_l)[C_l - \sum_{s \in S(l)} x_s]\right)\frac{d}{dt}p_l$$

$$\frac{d}{dt}p_l = \gamma(\sum_{s\in S(l)} x_s - C_l),\ \gamma > 0$$

$$Thus,\ \frac{d}{dt}D(p) = -\gamma\frac{d}{dp_l}f(p_l)[C_l - \sum_{s\in S(l)} x_s]^2$$

Since, $f^{'}(p_l) > 0$ and $\gamma > 0$ we can establish that *D(p_l(t))* is a decreasing function in t. Also since D is convex (see Proposition 4), there exists a minima, and $\frac{d}{dt}D(p(t)) \leq 0$ implies convergence to the optimal point.

C. Proposition 3: Under Assumptions (A1, A2) $\nabla D(p)$ is Lipschitz.

Proof: Define by *A* the incidence matrix where A_{ls} is 1 if source *s* uses link *l* and 0 otherwise. Further let the total number of links used by any source be bounded by *L* and the total number of sources by *S*. Then using equations (16,21) we can get the following (after some simplification)

$$\frac{\partial x(p)}{\partial p} = diag\left(\frac{1}{U_{obj}^{''}(x_s(p))}\right)A^T \qquad (22)$$

Also from equation (12) we get $\nabla D = f^{'}(p)(c - Ax)$. Differentiating it again with respect to p_l we get

$$\nabla^2 D = f^{''}(p)(c - Ax) + f^{'}(p)(-A\frac{\partial x(p)}{\partial p}) \qquad (23)$$

After some simplification $f^{''}(p)$ can be calculated as

$$f^{''}(p) = \frac{U_s^{3'}(x(p))}{\alpha_s^2} + \frac{\alpha_s}{U_{obj}^{3'}(x(p))} \qquad (24)$$

Since the Utility functions are strictly increasing in their arguments hence they will be rightly skewed, i.e. $U_s^{3'}$ is bounded away from 0. Further since the rate are bounded by *I* (Assumption A1) the second derivative of *f(p)* will be bounded, let us say that this bound is *F*. After some simplification the bound on $f^{'}(p)(-A\frac{\partial x(p)}{\partial p})$ can be calculated as βLS (for some $\beta > 0$ and β function of α_s). Using the capacity constraint we conclude that ∇D will be Lipschitz with the following bound

$$\|\nabla D(q) - \nabla D(p)\| \leq (FC + \beta LS)\,\|q - p\|$$

D. Proposition 4: Under assumption (A1, A2) D(p) is lower bounded, continuously differentiable and convex.

Proof: By Assumption (A1), U_s is bounded and continuously differentiable thus $U_s^{'-1}$ (and f(p)) exist and is also bounded and continuously differentiable. Therefore, D(p), as defined in equation (12) is also lower bounded and continuously differentiable.

Further from the assumption A1 (strictly increasing and strictly concave utility function) and equation (5) $f^{'}(p)$ (as defined in equation (24)) will be greater than 0. Using this knowledge and the capacity constraint the first term in equation (23) is always greater than or equal to 0. The second term of equation (23) is

$$f^{'}(p)(-A\ diag\left(\frac{1}{U_{obj}^{''}(x_s(p))}\right)A^T)$$

also strictly positive because from Proposition 1, $f^{'}(p)$ is always greater than or equal to 0, the incidence matrix is a 0-1 matrix and the utility functions are strictly concave. Thus we can say that equation (23) is greater than or equal to 0. Or $\nabla^2 D(p) \geq 0$ and *D(p)* is convex.

E. Proof of Theorem 1

By Propositions 3 and 4 the dual objective function *D(p)* is convex, lower bounded and $\nabla D(p)$ is Lipschitz, then any accumulation point p^* of the sequence $\{p(t)\}$ generated by the gradient projection algorithm is dual optimal [7]. Moreover, the constraints are linear and the primal problem is strictly concave hence there is no duality gap. Therefore dual optimal is also primal optimal.

REFERENCES

[1] D. Bansal and H. Balakrishnan. Binomial Congestion Control Scheme. *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[2] K. Chandrayana, Y. Xia, B. Sikdar and S. Kalyanaraman. A Unified Approach to Network Design and Control with Non-Cooperative Users. ECSE Techreport, ECSE-NET-2002-1, Mar 2002.

[3] 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, 1999.

[4] S. Floyd, etal. Equation-Based Congestion Control for Unicast Applications. In *Proc. of ACM SIGCOMM*, Aug 2000.

[5] Frank Kelly, Aman Maulloo and David Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the O.R. Society*, 49 (1998) 237-252.

[6] S. Kunniyur and R. Srikant. End-To-End Congestion Control: Utility Functions, Random Losses and ECN Marks., *Proc. of IEEE INFOCOM* , Tel-Aviv, Israel, March 2000.

[7] S. H. Low, D. E. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 7(6):861-75, 1999

[8] S. H. Low. A Duality Model of TCP and Queue Management Algorithms. *Proc. of ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, September 18-20, 2000, Monterey, CA.

[9] Packeteer. http://www.packeteer.com

[10] J. Padhye and S. Floyd. On Inferring TCP Behavior. In *Proc. of ACM SIGCOMM*, Aug 2001.