Network Fairness for Heterogeneous Applications

D. M. Chiu Adrian S. W. Tam
Department of Information Engineering
The Chinese University of Hong Kong
{dmchiu,swtam3}@ie.cuhk.edu.hk

December 30, 2004

Abstract

In this paper, we revisit the definition of fairness and TCP-friendliness in a network with heterogeneous applications. We suggest a utility optimization approach in which elastic and inelastic flows have different utilities. The fairness and TCP-friendliness are then evaluated based on a stochastic model of elastic and inelastic flows with finite files size, holding time and arrival rates (instead of the traditional model of flows competing indefinitely).

We argue that this view will open up new ways for elastic and inelastic flows to share the network, leading to improved performance for both kinds of traffic. Using this approach, we proceed to demonstrate distributed admission control can be used by inelastic flows instead of congestion control, and how this satisfies the new TCP-friendly criteria.

1 Introduction

Without special configurations, the Internet treats all packets in the same way, thus it provides a connectionless service. When there is congestion, TCP flows run a distributed congestion control algorithm that at least theoretically converge to fair bandwidth allocation based on the AIMD analysis [1]. Other applications may either use RTP [2] or use their own transport services that run over UDP. These flows typically grab whatever bandwidth they can manage to and do not worry about fair bandwidth allocation among competing flows.

There are therefore two classes of Internet citizens: TCP flows and UDP flows. The former types are considered "courteous" or "civilized" by following some conventions of behavior that are accepted as guaranteeing fair resource allocation during congestion. The latter types are considered "rude" network citizens who are not concerned about others. It is feared that this situation may be unfair to TCP flows and lead to congestion breakdown as the percentage of the UDP traffic increases. This is the well-known TCP/UDP co-existence problem.

A compromise solution was proposed: TCP-friendly congestion control. The applications using UDP (and RTP) often cannot respond to congestion without reducing their user satisfaction. The compromise requires these UDP applications to react to congestion more slowly. If all flows continue indefinitely and congestion persists, then a congestion control algorithm is considered TCP-friendly if it gradually brings the rate of UDP flows to the same as the TCP flows. It is a compromise because it lets these flows react slowly, but eventually these flows must converge to their fair share. What if the fair share is not adequate for a UDP application? The answer is Internet applications must be adaptable, i.e., to use whatever fair share bandwidth that the network gives them. For example, a video application may adapt by using lower resolution, fewer frames per second, or a smaller display area.

A significant number of efforts went into designing TCP-friendly congestion control algorithms[1] [3] and network adaptation mechanisms for multimedia applications [4]. In effect, the introduction of the "TCP-friendliness" notion broadens the range of network behaviors that are considered "civilized".

The question is, is the "TCP-friendly" framework accepted by the UDP applications? Our observation is that it is not, or at least not universally. The reason is that some applications may not be adaptable, or it is hard from them to adapt without sacrificing quality or ease-of-use. These flows from these applications have traditionally been referred to as *inelastic* flows.

The whole thesis of this paper is to suggest that we should further broaden the range of "civilized" behavior. We do this by redefining what is "TCP-friendly". We use the scenario where all flows run TCP congestion control as a benchmark. We then introduce reasonable performance metrics for TCP and UDP flows respectively — think of these metrics as utility functions. If a new control algorithm for UDP flows that results in a higher utility for both TCP and UDP flows for different traffic mixes (characterized by a model of arrival and departure rates, file sizes and playback rates and other parameters) is introduced, then the new control algorithm is a fair game. This methodology, if accepted, represents a paradigm shift in how we think of network fairness for heterogeneous applications.

As a concrete example, we show that some form of

[1]More discussion and references are given in section 5

admission control (instead of TCP-like congestion control) for UDP flows should be considered as TCP-friendly under our new definition. Adopting admission control for inelastic flows is hardly a new idea. In particular, Kelly et al [5] proposed and analyzed distributed admission control algorithms. Our new contribution in this paper is in making the case for reconsidering what is "TCP-friendly", and describing a methodology for judging whether a certain network behavior (such as admission control) is "TCP-friendly".

The organization of the paper is as follows. In Section 2, we present our new definition of network fairness and TCP-friendliness. In Section 3, we introduce a stochastic model and explain of how TCP-friendliness is evaluated for a particular example. That is, inelastic flows apply (distributed) admission control instead of TCP-like congestion control. In Section 4, we provide numerical results and conclude a suitable form of admission control is indeed TCP-friendly under the new definition. In Section 5, we briefly discuss related works. Finally, in Section 6, we give our concluding remarks and discuss directions for future work.

2 Fairness and TCP-friendliness Redefined

Let there be two classes of network flows: elastic flows that practice TCP congestion control and inelastic flows. Each elastic flow transfers a file whose size is randomly distributed. Its goal is to finish the transfer in as little time as possible. Each inelastic flow, on the other hand, needs to use the network for a randomly distributed period of time (known as *holding time*) at a constant bit rate.

The extent to which the needs for each type of flows are satisfied is measured by their respective utility functions $U_e()$ and $U_i()$. Suppose a flow arrives at time t_0 and is allowed to transmit at rate $x(t), t > t_0$. The function $x(t)$ is the service curve for this flow. The utility functions U_e and U_i can each be thought of as a function of $x(t)$ [2]. The social welfare W is the sum of the utility of all flows over a long time horizon

$$W = \sum_k U_e(x_k) + \sum_h U_i(x_h) \tag{1}$$

where k and h indexes respectively over all elastic and inelastic flow that arrive or complete during the given time horizon.

In recent literature, the thesis of [6] in redefining the fairness of bandwidth allocation in terms of the solution to utility optimization has been widely accepted. We therefore follow this convention by considering the set of service curves x_k and x_h that maximizes the social welfare and still satisfy the network capacity constraint to be the *fair* allocation. Semantically, however, there is a significant difference. Since both kinds of flows arrive at random times and have finite duration, fairness is not rigorously defined and achieved necessarily among flows that share the congested network simultaneously, but rather by flows that may use the network at different times.

There is a problem with applying equation (1) in practice. While the utility of the same type of users is assumed additive routinely, adding the utility of different types of users would require the different utility functions to be compatible and calibrated, which is not easy (or even possible) to accomplish in practice.

To appreciate the situation, let us consider an example. Let there be two flows, one elastic and one inelastic, sharing some common resource. The service curves achieved under control algorithm a are $x_1^a(t)$ and $x_2^a(t)$ for the elastic and inelastic flows respectively. Under a different control algorithm b, the achieved service curves are $x_1^b(t)$ and $x_2^b(t)$ respectively. Suppose the given utility functions $U_e()$ and $U_i()$ result in

$$U_e(x_1^a) + U_i(x_2^a) > U_e(x_1^b) + U_i(x_2^b)$$

Can we conclude algorithm a is better than (fairer than) algorithm b? This is highly questionable. Let us scale the utility function for the inelastic flow by a factor c, to reflect the fact that we cannot accurately calibrate the two utility functions. Then it is easy to see that the conclusion of whether algorithm a or b is better can flip depending on the value of c. In economics, utility functions are *ordinal* but not *cardinal*. In other words, it is reasonable to compare two situations based on their respective utility functions, for example, $U_e(x_1^a) > U_e(x_1^b)$; but it is dangerous to add the utility functions, especially for different types of users, and then compare them, as the above example shows.

To make our problem tractable, we do assume the utility of the same type of users can be added together. Therefore, we rewrite the social welfare as

$$W = W_e + W_i$$

where W_e and W_i denote the sum of utilities of the elastic flows and inelastic flows respectively. When comparing the total welfare achieved by control algorithms a and b, we require that both $W^a > W^b$ as well as the same relationship to hold for the individual components ($W_e^a \geq W_e^b$ and $W_i^a \geq W_e^b$) in order to conclude that algorithm a is greater than b.

In this paper, we are not seeking the optimal solution to the above utility optimization problem. Rather, we are interested in the comparison with using TCP for both elastic as well as inelastic flows, which is assumed to yield the following utility

$$W^{TCP} = W_e^{TCP} + W_i^{TCP}$$

When the inelastic flows use an alternative algorithm, a, then the resulting stochastic process would yield different

[2]That is, mapping the function $x(t)$ to a value. Sometimes, such utility function is known as a *functional*.

utility for both the elastic and elastic flows, to be denoted

$$W^{TCP+a} = W_e^{TCP+a} + W_i^{TCP+a}$$

We say the alternative algorithm *a* is *TCP-friendly* if

$$W_e^{TCP+a} \geq W_e^{TCP} \quad (2)$$

On the other hand, the alternative strategy should also make sense for the inelastic flows, in the sense

$$W_i^{TCP+a} \geq W_i^{TCP} \quad (3)$$

Together, we also have

$$W^{TCP+a} \geq W^{TCP} \quad (4)$$

So our new definition of TCP-friendliness is really based on all the above properties (in equations 2, 3 and 4) being true.

Another way to think of $W_e()$ and $W_i()$ is to consider them as some system performance metrics which are used to measure the outcome for an aggregated class of users. We use TCP congestion control to establish the benchmark performance target. We want to find alternative *robust* control algorithms that out-perform TCP for inelastic flows, for various system parameters.

What we have introduced and discussed in this section is a general framework for redefining network fairness for heterogeneous applications. In the next section, we will introduce specific models and evaluate the TCP-friendliness of specific control algorithms.

3 Modeling and Evaluating the Fairness of Admission Control

We now introduce a specific model with several simplifying assumptions to make our problem tractable.

Let us assume that the elastic and inelastic flows are stationary stochastic processes with Poisson arrival rates λ_e and λ_i. The network is a single bottleneck and its bandwidth is 1. The elastic flows have exponentially distributed file sizes with mean equal to $1/\mu_e$. So, if the entire bandwidth is used to serve an elastic flow, it has a departure rate of μ_e. The inelastic flows have exponentially distributed holding times with mean $1/\mu_i$ and a playback rate of $\alpha \leq 1$. So, once an inelastic flow starts, it consumes α of the bottleneck's bandwidth and departs at the rate of μ_i.

The elastic flows are always assumed to be implementing TCP congestion control which consumes the bottleneck bandwidth (available to them) in equal proportions. For inelastic flows, we are interested in evaluating the following different strategies:

1. Perform TCP-friendly congestion control and split the bottleneck bandwidth equally among TCP flows when there is a congestion. When the fair share for each flow is greater than α, the inelastic flows would still consume α [3]. Otherwise, it is assumed that the convergence to fair share is instantaneous which is the same as the elastic flows' controls. This is the benchmark case. We refer to this simply as the TCP strategy.

2. Perform neither congestion control nor admission control. This models the behavior of today's UDP flows. In this case, when $1/\alpha$ or more inelastic flows are in the network, the network capacity is totally consumed (equally shared) by the inelastic flows. Therefore, the elastic flows get no service. Otherwise, if there are $m < 1/\alpha$ inelastic flows, each inelastic flow gets α and the elastic flows share the remaining $(1 - m\alpha)$. We refer to this as the UDP strategy.

3. Perform "selfish" admission control but no congestion control. In general, the admission control function can be quite sophisticated. For example, it may depend on the playback rate α and the holding time μ_i. In our analysis, we consider two simple admission control strategies, in each case assumed to be performed by the arriving flow itself. If the arriving inelastic flow is "selfish", it admits itself if

$$n\varepsilon + (m+1)\alpha \leq 1$$

where (n,m) is the current number of elastic and inelastic flows in the network. Here, ε represents the minimum rate consumed by an elastic traffic already in the network. Typically, $\varepsilon \ll \alpha$. In other words, the selfish flow admits itself as long as it is possible to achieve its playback rate of α, even though it means all elastic flows have to run at their minimum rate of ε. We refer to this as the AC1 strategy.

4. Perform "considerate" admission control but no congestion control. In this case, the arriving inelastic flow (assumed to be "considerate") admits itself if

$$\frac{1 - m\alpha}{n+1} \geq \alpha.$$

In other words, the considerate flow assumes hypothetically that it is an elastic flow, and admits itself only if its fair share of the bandwidth is no smaller than α. The above condition actually reduces to

$$(n+m+1)\alpha \leq 1.$$

We refer to this as the AC2 strategy.

Further, both the elastic and inelastic flows and their controls are modeled using fluid approximation. This means all the flows share the network as a processor sharing server, and all the control regulating the flows take effects immediately. Given these assumptions, each of the four scenarios above can be modeled by a Markov Chain with two-dimensional state space, as shown in Figure 1.

[3]This is a different treatment than the model in [7]

		$(n,m) \to (n,m+1)$	$(n,m) \to (n+1,m)$	$(n,m) \to (n,m-1)$	$(n,m) \to (n-1,m)$
TCP	$(n+m)\alpha \leq 1$	λ_i	λ_e	$m\mu_i$	$(1-m\alpha)\mu_e$
	$(n+m)\alpha > 1$	λ_i	λ_e	$m\mu_i$	$\frac{n}{n+m}\mu_e$
UDP	$m\alpha \leq 1$	λ_i	λ_e	$m\mu_i$	$(1-m\alpha)\mu_e$
	$m\alpha > 1$	λ_i	λ_e	$m\mu_i$	0
AC1	$n\varepsilon + (m+1)\alpha \leq 1$	λ_i	λ_e	$m\mu_i$	$(1-m\alpha)\mu_e$
	$n\varepsilon + (m+1)\alpha > 1$	0	λ_e	$m\mu_i$	$\max(0,(1-m\alpha)\mu_e)$
AC2	$(n+m+1)\alpha \leq 1$	λ_i	λ_e	$m\mu_i$	$(1-m\alpha)\mu_e$
	$(n+m+1)\alpha > 1$	0	λ_e	$m\mu_i$	$\max(0,(1-m\alpha)\mu_e)$

Table 1: State transition rates corresponding to the four Markov models

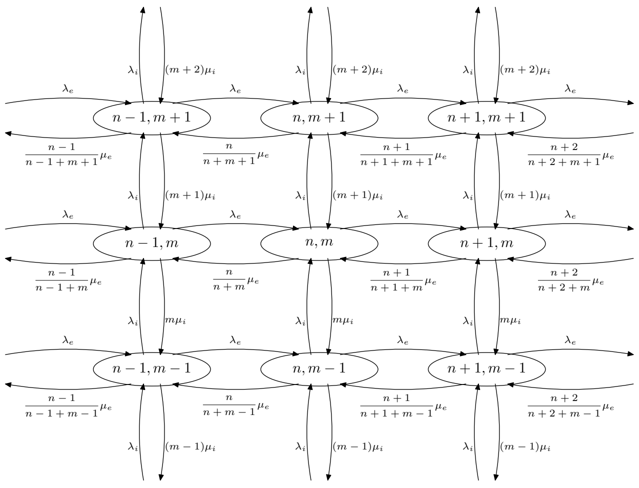

Figure 1: Markov model of two type of flows with different controls

The Markov transition rates of the four scenarios are summarized in Table 1.

Independent of what control strategy is to be used for inelastic flows, the arrival rate of elastic flows is always λ_e since there is no admission control for elastic flows. For inelastic flows, the arrival rate is always λ_i except for the two cases (AC1 and AC2) where admission control is in effect. The inelastic flows are always departing at a rate of μ_i which is their average holding time. The service rate allocated to the elastic flow, however, is different for the four different cases. When the inelastic flows use TCP control, the elastic flows will get at least their fair share, or more if the inelastic flows cannot use theirs. For the other cases (AC1, AC2 and UDP), the inelastic flows will not perform congestion control. Hence, the elastic flows will only get what is not used by the inelastic flows $(1-m\alpha)$, or zero if $1 \leq m\alpha$.

These Markov models generally do not have simple product form solutions, except for certain boundary cases. For example, in the AC1 case, if ε is zero, then the state transitions for inelastic users are independent of the elastic users and the system has a product-form solution. In fact for that case, the blocking probability for inelastic users is given by the well-known Erlang-B formula.

The steady state probability distribution $P(n,m)$ can be

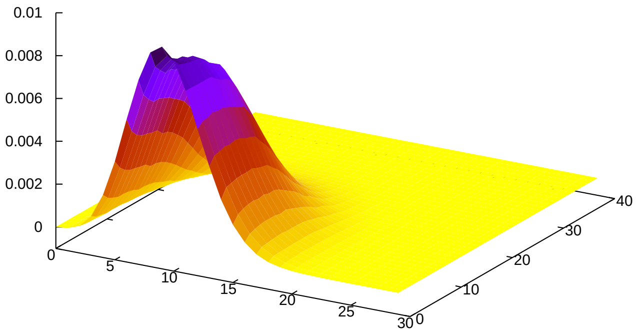

Figure 2: Probability $P(n,m)$ when $\rho = 0.95$, with $\rho_e = \alpha\rho_i$ and $\alpha = 0.05$

derived from simulation. Due to the Markov assumptions, the Markov chain itself (see Fig 1) is simulated for a variety of parameters. We validated the simulation tool by verifying the results with those cases where we have closed-form solutions.[4] Figure 2 shows a typical probability density function $P(n,m)$ obtained from one of our simulations using the TCP strategy with $\rho = 0.95$.

The other difficult task in this modeling exercise is to define some plausible utility functions for both the elastic and inelastic flows. We have some intuition of what we want:

- For inelastic flows, users care about whether the desired playback rate, α, is achieved at every moment of the holding time.
- For elastic flows, users want each bit of the file to be transferred as fast as possible.

Additionally, we want the utility functions to be normalized so that:

- The amount of utility achieved per unit time for one unit of the bandwidth is between zero and one.

To make the evaluation tractable, we let both utility functions be decomposable. For elastic flows, we define

[4] We also built a simulation tool to for the queueing system, which can be used to investigate non-exponential interarrival and service times. This is significant slower, but served as another way to validate the Markov chain simulation.

the utility for the transfer of each bit of information as a log function of the rate at which the bit is transferred.

$$v_e(r) = \ln(1+(e-1)r)$$

where e is the base of natural logarithm.

For flow k with file size s, transferred at rates (determined from the service curve, x_k) $r_1, r_2, ..., r_s$, the total utility of the flow is

$$U_e(x_k) = \sum_{1 \le i \le s} ln(1+(e-1)r_i) \qquad (5)$$

The total utility of a large number of elastic flows over a long time horizon can then be approximated as the product of arrival rate of elastic flows and the utility of an average elastic flow in the steady state:

$$\begin{aligned}\sum_k U_e(x_k) &\approx \lambda_e \int_0^\infty s_k \mu_e e^{-\mu_e s_k} ds_k \sum_{n \neq 0, m} v_e\left(a_e(n,m)\right) P(n,m|n \neq 0) \\ &= \rho_e \cdot \sum_{n \neq 0, m} v_e\left(a_e(n,m)\right) P(n,m|n \neq 0) \\ &= \rho_e \overline{v_e}.\end{aligned}$$

where $a_e(n,m)$ represents the allocation of bandwidth to an elastic flow when the system is in state (n,m). Note that the average rate of a file transfer is computed over all conditional state probabilities where at least one elastic user is in the system. We call $\overline{v_e}$ the average utility generated by each bit of elastic traffic transferred.

Similarly for inelastic users, we define the utility per unit time, $v_i(r)$, as a normalized arctan function that equals to one when the allocation is α or higher, but drops quickly to zero for lower allocated rates.

$$v_i(r) = \begin{cases} \frac{1}{\pi} \arctan\left(\gamma(r - \beta\alpha)\right) + \frac{1}{2} & r < \alpha \\ 1 & r \ge \alpha \end{cases}$$

This function is monotonically increasing in r. When $r = \beta\alpha$, $v_i(r)$ rises to $\frac{1}{2}$. So the parameter β controls the transition point from low to high utility values. The parameter γ controls how sharp the transition is. For large values of γ, the function approaches a step function.

We assume that the utility of each inelastic flow is v_i summed (integrated) over the holding time. Thus the utility of the h^{th} user who has the holding time T_h is:

$$U_i(x_h) = \alpha \frac{1}{T_h} \int_{T_h} v_i(x_h(t)) dt. \qquad (6)$$

Here, α serves as a normalizing constant, chosen based on the fact that an inelastic user uses up only α of the bandwidth at most. Since we compare elastic and inelastic utilities separately (as discussed in the last section), such scaling does not affect our conclusions.

The total utility of inelastic flows over a long time horizon can again be approximated by the product of the arrival rate of inelastic flows, λ_i, with the non-blocking probability, $(1-B)$:

$$\begin{aligned}\sum_h U_i(x_h) &\approx \lambda_i (1-B) \alpha \frac{1}{\mu_i} \sum_{n,m \neq 0} v_i(a_i(n,m)) P(n,m|m \neq 0) \\ &= (1-B)\alpha\rho_i \overline{v_i}.\end{aligned}$$

We call $\overline{v_i}$ the average utility generated by each bit of inelastic traffic.

In our following discussions, the default parameters used are $\gamma = 1000$ and $\beta = 0.9$, unless noted otherwise. The normalized ln() and arctan() functions with these parameters are plotted in figure 3 for reference.

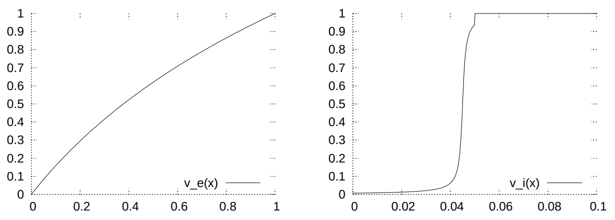

Figure 3: Elastic (left) and inelastic (right) utility at different rate

The decomposability of the utility function is a strong assumption. It assumes the treatment received by different bits are independent of each other. For example, one might argue that if the service curve of an inelastic flow drops below α often enough, then it becomes a more severe negative factor. However, the adoption of this assumption affords us much simplified analysis. While we lose some generality of the utility functions, we gain in the ability to study the robustness of parameters.

To summarize, our model consists of two separate parts. One deals with the stochastic process of the elastic and inelastic flows sharing a network, based on which we can compute (or simulate) the state probabilities $P(n,m)$. The other part deals with the utility of elastic and inelastic flows. Because of the decomposability assumption on the utility functions, these two parts can be computed separately as shown in Figure 4. Once the utilities are computed, we can whether various control algorithms are TCP-friendly by checking the conditions specified in equations (2), (3) and (4).

4 Numerical Results

4.1 Summary of Parameters

We report our simulation results in this section. Table 2 summarizes the important parameters in our model.

In our model, the total offered load by elastic flows is

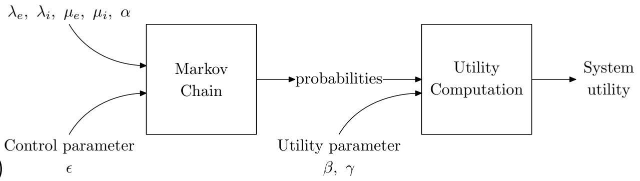

Figure 4: The parameters and processes of computing utilities

given by

$$\rho_e = \frac{\lambda_e}{\mu_e},$$

and the corresponding offered load by inelastic flows is

$$\alpha\rho_i = \frac{\alpha\lambda_i}{\mu_i}.$$

So, the total offered load is

$$\rho = \rho_e + \alpha\rho_i.$$

Symbol	Parameter Explanation	Default
λ_e	arrival rate of elastic flows	
λ_i	arrival rate of inelastic flows	
μ_e	departure rate of elastic flows if bottleneck bandwidth is used exclusively for its service	
μ_i	inverse of the average holding time for inelastic flows	
α	desired throughput for inelastic flows (also known as playback rate)	0.05
ε	the minimum fraction of the bottleneck bandwidth consumed by an elastic flow once it is started	0.001
γ	a parameter that controls the steepness of the inelastic flow's utility function	1000
β	a related (to γ) parameter in $v_i()$	0.9

Table 2: Model parameters and their default values

4.2 The Congested Regime

When the network is lightly loaded, i.e., $\rho \ll 1$, we expect that it does not matter which control algorithm we use for inelastic flows — they should all yield similar results. Our interest is to evaluate the performance of the different strategies in the congested regime.

First, it appears that the congested regime is when ρ is close to but less than 1. Upon closer examination, the systems except UDP can still be stable[5] when ρ is greater than 1, as long as $\rho_e < 1$. This is a direct result of how inelastic flows are defined. Since the holding time, $1/\mu_i$, remains the same even when the inelastic flows receive less than their desired playback rate, α, the effective load due to inelastic flows automatically adjusts downwards as ρ exceeds 1 when TCP is used to control the inelastic flows. When the admission control algorithms are adopted (with non-zero ε for AC1), the system is also stable as ρ exceeds beyond 1. Therefore, for the congested regime, we consider cases when ρ well exceeds 1.

4.3 Comparison for Different Offered Load

The first set of figures compares the average total elastic and inelastic utility for increasing total offered load ρ. For this experiment, the ratio of elastic and inelastic offered load is 1, i.e., each contributes 50% of the traffic load.

[5]The number of flows in the system stays bounded.

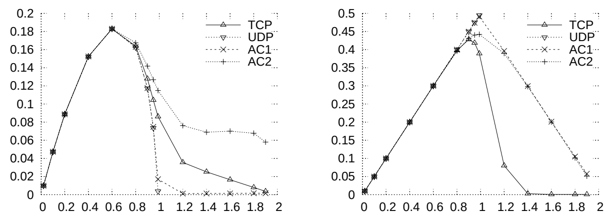

Figure 5: Elastic (left) and inelastic (right) utility vs. ρ, with $\rho_e : \alpha\rho_i = 1 : 1$

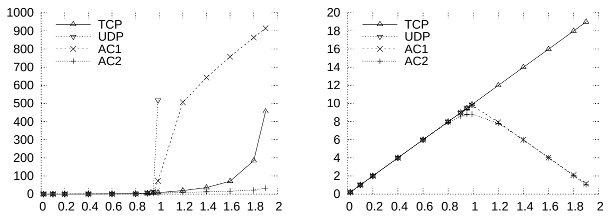

Figure 6: Mean population of elastic (left) and inelastic (right) flows vs. ρ, with $\rho_e : \alpha\rho_i = 1 : 1$

As shown in Figure 5, the general pattern is that the normalized average utility generated by the system in steady state increases with offered load until the offered load reaches the network capacity (assumed to be 1). When the network is subjected to more than what it can handle, its service starts to degrade, and results in reduced utilities. First, let us observe the inelastic traffic. For $\rho < 1$, all strategies perform roughly the same. For $\rho > 0.95$, if TCP is used, the utility for elastic flows drops very sharply to zero. This is because TCP would give these elastic flows their fair share, and at such high offered load, the fair share generates almost zero utility. For the UDP case, since it does not yield at all to other traffic, the system becomes unstable after the offered load reaches 1. In the congested regime ($0.95 < \rho < 1$), the inelastic flows still achieve high utility whereas the elastic flows get almost none.

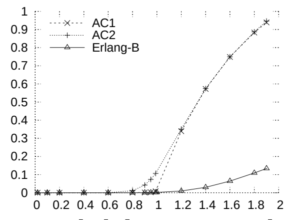

Figure 7: Blocking probability vs. ρ, with $\rho_e : \alpha\rho_i = 1 : 1$

Even when admission control is applied, the utility of the inelastic flows drops steadily as ρ increases beyond 1, since the steady state number of elastic flows grows and takes up an increasing amount of bandwidth. The mean population sizes for elastic users are plotted in Figure 6. The figure shows that the population of elastic users in the AC1 case rises much faster than that in the AC2 case. This leads to similar blocking probabilities as plotted in Figure 7. The Erlang-B blocking probability is also included in the figure as a reference curve. The Erlang-B formula provides a good estimate for AC1 and AC2 when $\rho \leq 1$. When $\rho > 1$, the blocking probability of both AC1 and AC2 depends on the number of elastic users in the system.

For elastic flows, the population size increases drastically when $\rho > 0.95$, especially for the UDP and AC1 cases (for the UDP case, the network is unstable when $\rho > 1$). For the TCP and AC1 cases, the service degradation is more graceful. Particularly encouraging is the fact that AC1 (the considerate admission control) yields the best result for elastic flows without sacrificing the services to inelastic flows.

4.4 Comparison for Different Traffic Mix

In the next set of figures 8 and 9, we compare the different strategies when the traffic mix (elastic versus inelastic) is 1:9 and 9:1.

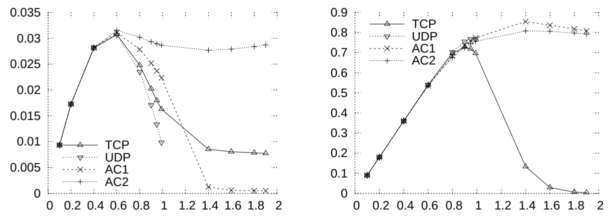

Figure 8: Elastic (left) and inelastic (right) utility vs. ρ, with $\rho_e : \alpha\rho_i = 1 : 9$

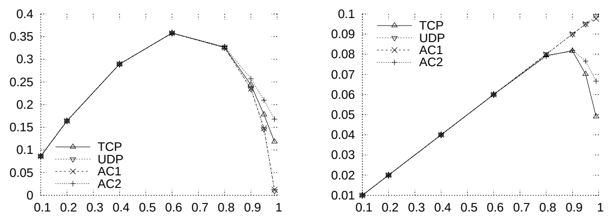

Figure 9: Elastic (left) and inelastic (right) utility vs. ρ, with $\rho_e : \alpha\rho_i = 9 : 1$

The general pattern is the same, i.e., the utility for both elastic flows and inelastic flows increases with the offered load. The role of the elastic offered load ρ_e is further illustrated. When 90% of the load is elastic, the model becomes unstable when the total offered load ρ reaches beyond 1, no matter which control strategy is used. On the other hand, if only 10% of the load is elastic, the network can operate quite satisfactorily when $\rho > 1$. In particular, the considerate admission control (AC2) is able to yield very little service degradation for $\rho > 1$. In contrast, TCP results in graceful service degradation for elastic traffic, but sharp degradation for inelastic traffic; AC1 results in good performance for inelastic flows but sharp service degradation for elastic traffic; UDP is similar to AC1 but is worse since the network becomes unstable when $\rho > 1$.

The encouraging observation is that the considerate admission control has out-performed TCP in all the situations so far.

4.5 Comparison for Different Playback Rate

It is understandable that, the biggest concern with exempting inelastic flows from applying congestion control is that they might have a large playback rate α. A few "large" inelastic flows may significantly impact the performance of elastic flows. Our intuition is that, a suitably designed admission control scheme would neutralize the damage by blocking judiciously some of these large flows at the right time.

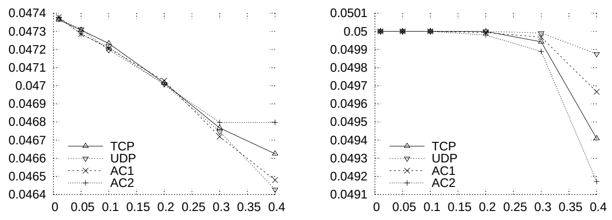

Figure 10: Elastic (left) and inelastic (right) utilities vs. α, with $\rho = 0.1$

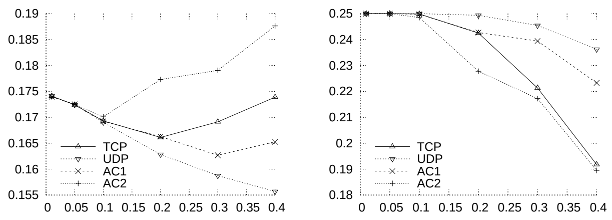

Figure 11: Elastic (left) and inelastic (right) utilities vs. α, with $\rho = 0.5$

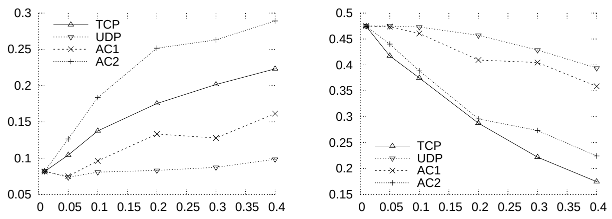

Figure 12: Elastic (left) and inelastic (right) utility vs. α, with $\rho = 0.95$

Figures 10, 11, 12, and 13 compare the elastic and inelastic utilities as we vary α for the light load ($\rho = 0.1$), medium load ($\rho = 0.5$), heavy load ($\rho = 0.95$) and heavily congested ($\rho = 1.4$) cases respectively.

For small values of α, or when the offered load is low as compared to the network capacity, the different strategies give similar performance. This is expected. As we increase α, for each of the load levels, the considerate admission control (AC2) improves the elastic utility when compares with TCP. For larger values of α under light or medium load (Figure 10 and 11), AC2 actually performs slightly worse than TCP for inelastic utility. This is because when the network can only accommodate a small number of inelastic users (when α is large), even at light load, there is a small but significant probability that some inelastic flows

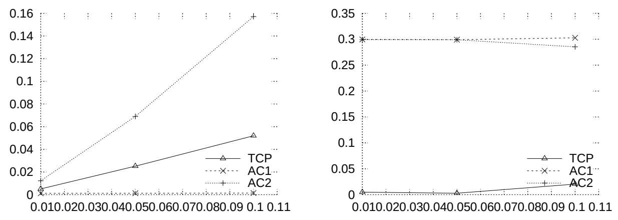

Figure 13: Elastic (left) and inelastic (right) utility vs. α, with $\rho = 1.4$

get blocked. We expect this problem can be minimized with suitable tuning and added sophistication to AC2.

As expected, both AC1 and UDP produce improvements for inelastic flows, but more degradation to elastic flows in comparison to TCP, which does not meet our requirements for TCP-friendliness. In Figure 13, UDP is not shown since it is not stable for $\rho > 1$.

Finally, we also did some experiments to look at the sensitivity to α when the traffic mix is unbalanced. As shown in Figure 14 and 15, when the traffic ratio is 9 : 1 or 1 : 9 (elastic vs. inelastic), the AC2 strategy still outperforms TCP for both elastic and inelastic utilities.

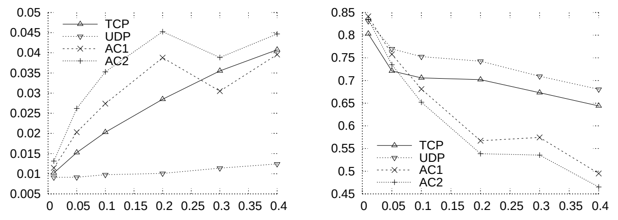

Figure 14: Elastic (left) and inelastic (right) utility vs. α with $\rho = 0.95$, $\rho_e : \alpha\rho_i = 1 : 9$

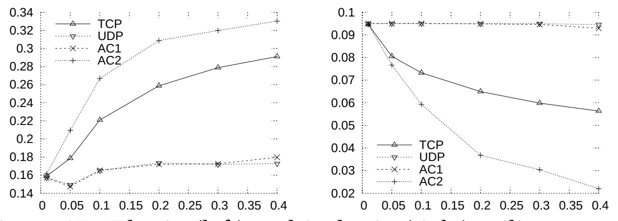

Figure 15: Elastic (left) and inelastic (right) utility vs. α with $\rho = 0.95$, $\rho_e : \alpha\rho_i = 9 : 1$

4.6 Sensitivity to Other Parameters

Could it be that the positive results work only for the specific pair of utility functions which we choose?

The normalized ln() function that we use for the elastic flow is fixed. The arctan() function that we use for the inelastic flow has a couple of parameters (γ and β) that control the shape of the function. We have repeated our experiments by trying $\gamma = 100$ and $\beta = 0.7$. Although the curves moved a little, the relative behaviors did not change much.

We also try to vary the ratio of λ_e to λ_i and μ_e to μ_i, while keeping the values of ρ_e and ρ_i to be the same. The results for most cases are very similar as before, indicating that the performance metrics are mostly dependent on ρ_e and ρ_i rather than their components. This indicates that there are likely simpler formulas (either closed-form solutions or performance bounds) that depend on ρ_e and ρ_i only. The exception is the TCP case, where the results vary slightly as we change the ratios. The solution for this case may be more complicated. In any event, none of those results changes any of the conclusion in the last few subsections.

5 Discussion of Related Work

Network fairness is always an issue when discussing resource allocation and congestion control [8, 9]. Kelly et al [6] linked fairness to network utility optimization, and it was further generalized in [10].

The Internet community, and IETF especially has always been interested in this topic, more from the view point of Internet operations and architecture [11, 12]. It is in this context the ”TCP-friendly” notion is fostered.

There is a large volume of work on various TCP-friendly congestion control ideas, for example [13, 14, 15, 16], to cite just a few. The purpose of this paper is really to point out a different way to define TCP-friendliness and to explore more ways for heterogeneous applications to co-exist on the Internet. The large body of work on the whole TCP-friendly congestion control approach is therefore only indirectly related.

Over the last few years, Roberts, Massoulie, Key, Kelly and others have made several important contributions to the analysis of bandwidth allocation for elastic and inelastic traffic [5, 7, 17, 18, 19, 20]. This is very close to what we are exploring. However, there are some differences. First, we seek to redefine the notion of TCP-friendly bandwidth sharing. Therefore, we need to establish the utility functions and the methodology for justifying that a control algorithm is fair. This aspect is not covered by the above body of work. Second, we model the inelastic flows in a slightly different way. Namely, the inelastic flows consume at most the playback rate and no more of the bottleneck bandwidth.

6 Conclusion and Future Work

In this paper, we propose a new direction in thinking about the fairness, and the notion of “TCP-friendliness” for a network with both elastic and inelastic flows. It is our hope that this new definition will lead to new distributed traffic control strategies that will allow both kinds of traffic to better co-exist.

We then use our new definition and methodology to study the suitability of using (distributed) admission control instead of congestion control for inelastic traffic. Although the results are preliminary (mostly based on simulation), they are very encouraging, suggesting that our new approach is very promising.

We think this is just the beginning of potentially many new efforts along this line of thinking. Some ideas for future work are listed below:

- How about applying admission control to elastic flows as well? As seen in Figure 5, and 9, the network utility drops sharply (or the network simply becomes unstable) as ρ_e increases and when $\rho > 1$. Admission control, when applied to elastic flows in some suitable manner, may also improve the situation under high offered load.
- In this study, the results are entirely generated by simulation. Although we consider a large number of cases (of different system parameters), we may not have considered all cases of interest. We plan to simulate some additional cases which we have not considered.
- We plan to derive performance bounds (and closed-form solutions in selected cases) to further consolidate the results. This helps to avoid simulations for all cases.
- In this study, we consider the case where the network is a single bottleneck link. We would like to extend the results to flows sharing works with arbitrary topologies.
- We use two different functions to represent the utilities of elastic and inelastic flows. We expect that it should be possible to use a single function with different parameters to represent the utility of the different type of users. In this way, we can more readily study the sensitivity in comparing different strategies to variations in the utility functions.
- We plan to implement the new control strategies as an extended socket interface and experiment with real applications.

References

[1] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, 1989.

[2] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport for real-time applications." IETF RFC3550, Jul 2003. Standard.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of SIGCOMM*, Aug 2000.

[4] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification." IETF RFC3448, Jan 2003. Proposed Standard.

[5] F. P. Kelly, P. B. Key, and S. Zachary, "Distributed admission control," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 2617–2628, 2000.

[6] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[7] P. Key, L. Massoulie, A. Bain, and F. Kelly, "A network flow model for mixtures of file transfers and streaming traffic," in *Proceedings of ITC 18*, 2003.

[8] R. Jain, D. M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC Technical Report DEC-TR-301, DEC, Sep 1984.

[9] D. Bertsekas and R. Galleger, *Data Networks*. Prentice Hall, 1987.

[10] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, 1998.

[11] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, 1999.

[12] S. Floyd and J. Kempf, "IAB concerns regarding congestion control for voice traffic in the internet." IETF RFC3714, Mar 2004. Informational.

[13] J. Mahdavi and S. Floyd, "TCP-friendly unicast rate-based flow control." Technical note sent to the end2end-interest mailing list, Jan 8 1997.

[14] J. D. Padhye, *Model-based Approach to TCP-friendly Congestion Control*. PhD thesis, University of Massachusetts Amherst, Sep 2000.

[15] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proceedings of ICNP*, 2000.

[16] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *Proceedings Sigcomm 2001*, 2001.

[17] L. Massoulie and J. W. Roberts, "Bandwidth sharing and admission control for elastic traffic," *Telecommunication Systems 15*, pp. 185–201, 2000.

[18] N. Benameur, S. B. Fredj, S. Oueslati, and J. Roberts, "Quality of service and flow level admission control in the internet," *Computer Networks*, vol. 40, pp. 57–71, 2002.

[19] L. Massoulie and J. Roberts, “Bandwidth sharing: Objectives and algorithms,” *IEEE/ACM Transactions on Networking*, vol. 10, 2002.

[20] T. Bonald and A. Proutiere, “On performance bounds for the integration of elastic and adaptive streaming flows,” *ACM Sigmetrics Performance Evaluation Review*, vol. 32, pp. 235–245, 2004.