BROADNETS 2008 - 5th Intl. Conf. Broadband Communications, Networks and Systems. London, UK. Sep 2008.

Revisiting inter-flow fairness

Saleem Bhatti, Martin Bateman
Devan Rehunathan, Tristan Henderson
& Greg Bigwood
School of Computer Science
University of St Andrews, UK
{saleem, mb, dr, tristan, gjb}@cs.st-andrews.ac.uk

Dimitrios Miras
Department of Computer Science
UCL, UK
d.miras@cs.ucl.ac.uk

***Abstract*—Many new transport protocols are being defined, including, for example, variants of the Transmission Control Protocol (TCP), to better match the requirements of new applications. A key issue in the evaluation of protocol flows, in terms of their performance, is how *fair* they are to other flows. Specifically, it is important to understand how a mix of existing and/or new protocols will interact with each other when using the same network resources. Such observations help to inform protocol design, and allow an assessment of potential impacts on users. We present a simple, yet effective, methodology for examining a specific case of inter-flow fairness based solely on measurements of flow performance. As well as using an existing fairness metric, we propose a new metric which provides a richer information summary for the evaluation of fairness.**

I. Introduction

The additive increase multiplicative decrease (AIMD) behaviour [10] of the Transmission Control Protocol (TCP) is often credited as a major factor in the stability of today's Internet. This causes TCP to backoff when it experiences congestion, cutting its transmission rate to half, and then only increasing its transmission rate by one segment size every round trip time (RTT). However, this conservative behaviour, coupled with TCP's mode of operation (reliable, ordered, byte-stream), is not necessarily suitable for all applications, and so researchers are interested in designing new protocols to meet new application requirements. A related area of interest is the high-speed operation of TCP. Specifically, the operation of TCP over network paths with a high bandwidth-delay product (BDP) causes slow recovery of transmission rates, after a congestion event. New variants of TCP have thus been defined so that this behaviour is improved on high-BDP paths.

TCP's behaviour allows flows to adapt their transmission rate in order to match network conditions. Consequently, traffic that does not behave in the same manner, and is more aggressive in its transmission behaviour, will cause TCP (and TCP-like) flows to back-off 'unfairly' when they share the same network path. To address this concern, in a best-effort Internet Protocol (IP) network (with no other policing or rate control in place), new and existing protocols should exhibit 'fairness' towards TCP flows[1] and towards each other so that they do not unfairly limit the rate of other flows.

[1]The notion of "TCP friendliness": http://www.psc.edu/networking/projects/tcpfriendly/

But how can we assess 'fairness' in the behaviour of such protocols? A popular measure of fairness for network flows is *Jain's Fairness Index (JFI)* [4]. While this is a general metric, by using the end-to-end throughput of the flows sharing (the whole or part of) a network path, system-wide (i.e., across all flows) fairness can be assessed. Starting with Jain's Fairness Index, then introducing a new metric – *Normalised Resource Usage (NRU)* – we propose a simple but practical methodology for examining the fairness of flows. Our key contributions are:

- a new metric with which to assess the dynamics of inter-protocol fairness, convergence time and stability.
- a measurement-based approach which allows the new metric to be used easily.

We first introduce the idea of *fairness* for flows (Section II). We then define our new metric (Section III) and describe our methodology for evaluation, including our testbed (Section IV). Section V presents our experiments, conducted as pair-wise tests, firstly examining intra-protocol behaviour – how flows of the same protocol interact with each other – and then Section VI demonstrates inter-protocol behaviour by examining the interaction between our choice of subject protocol (DCCP/CCID2) and two other protocols (NewReno and CUBIC). We conclude with a short discussion of the limitations of our approach (Section VII).

II. Assessing inter-flow fairness

The Transport Modelling Research Group (TMRG) of the IRTF[2] presents the criteria by which one might make rigorous and complete assessments, notably comparative assessments, of transport protocols [5]. The following are noted (amongst others) by the TMRG as desirable metrics for assessing the performance of transport flows:

- *inter-flow fairness:* how are resources shared between different flows?
- *stability:* is the protocol behaviour stable, or given to oscillatory behaviour?
- *convergence time:* how long does it take, after some epoch, for flows to converge to fairness (at least to within some tolerance value of fairness)?

Our intention is to demonstrate a methodology for measuring inter-flow fairness that is easy to implement. We show how

[2]http://www.icir.org/tmrg/

fairness, as evaluated using end-to-end performance measurements (we chose throughput), can give simple and practical assessments of stability and convergence time.

A. What is fair?

The notion of 'fairness' in the use of resources has been much debated within the literature. Having a fair share of a resource is important where the resource demands of multiple flows sharing the resource are not met. In the absence of any other resource controls in the network, this means that there is at least one point along the end-to-end path where congestion is occurring, and we may determine how the resource is being shared by evaluating the resource distribution across the flows on that (part of) the path. For example, in the case of transport protocol flows sharing a bottleneck link on an end-to-end path, we could evaluate the way that the capacity is shared at the bottleneck (a local view), or the end-to-end throughput achieved for each flow sharing the bottleneck (a global view).

B. Definitions of fairness

What is a *fair* share of a resource? There are several well-known definitions of fairness, and we take the list below from the work of the TMRG.

In *max-min* fairness [11], each flow's throughput is at least as large as that of all other flows which have the same bottleneck. In this scheme each flow's demand is met, with the minimum demand (request for allocation) achieving the maximum allocation of resource. This assumes that the flow's demand is known, or (in the absence of this knowledge or no other resource usage model), that all flows effectively receive an equal share of the resource.

The goal of *proportional fairness* [12] is to maximise the utility function $U = \sum_1^N \log T_n$ for a given set of N flows, where T_n is the throughput of flow n. However, the implicit assumption that the utility can be modelled as a log function has not been justified.

Of course, *weighted* versions of max-min and proportional fairness are also possible, to reflect, for example, different assignments of capacity. With both max-min fairness, and proportional fairness, there is also the assumption that resource allocation can be controlled. In a best-effort IP network, we are typically unable to control resources, but we may be able to *measure* usage of resources, especially (but not exclusively) at end-systems. In our aim to create a simple and practical methodology for assessing fairness, it would be beneficial to use a metric that is easily facilitated through some measurement related to a flow's performance, e.g. measurement of a flow's end-to-end throughput.

Whilst the metrics listed above focus on throughput, other proposals suggest using different measures to assess fairness. For example, there are proposals to use end-to-end delay for file transfers [14], or some notion of 'cost' (e.g., [3]). We choose to use end-to-end throughput, as it remains applicable to assessment of flow performance, is widely used, is easily understood and is straightforward to measure. However, our metric is general and could also be applied using end-to-end delay or cost, if required.

As mentioned above, Jain's Fairness Index (JFI) [4] is widely used for assessing *system-wide* fairness, as in Equation (1), where, $0 \leq J \leq 1$, N is the number of flows, r_n is the value of the resource attribute being assessed for flow n, e.g. r_n is the measured end-to-end throughput. $J = 1$ means there is fairness across all flows; $J = 0$ indicates no fairness.

An obvious approach to examining system-wide fairness over time is simply to evaluate the JFI at given instances during the period of interest, as in Equation (2), where t, in practice, is discrete. $r_n(t_j)$ is then an approximation of throughput as determined at time interval t_j, and evaluated over the period $(t_j, t_{j-1}), t_j > t_{j-1}$, where t_{j-1} is the previous time at which an approximation was determined. For our experiments, t was every second. So, J is the mean value of $J(t)$ values over a given time period.

$$J = \frac{\left(\sum_{n=1}^{N} r_n\right)^2}{N \sum_{n=1}^{N} r_n^2} \tag{1}$$

$$J(t) = \frac{\left(\sum_{n=1}^{N} r_n(t)\right)^2}{N \sum_{n=1}^{N} r_n(t)^2} \tag{2}$$

The definition of JFI means that it may be difficult to determine the degree of relative unfairness between the flows. To illustrate, in Figure 1 we have created an artificial situation with two flows. Flow 2 is held constant at 100 and the value of Flow 1 is varied from 1 to 10000 (the units are immaterial). The plot shows the value of JFI (Equation 1) as the ratio Flow 1 / Flow 2 changes: the ratio has a range of four orders of magnitude, whilst the JFI has the range $[0.51, 1.00]$.

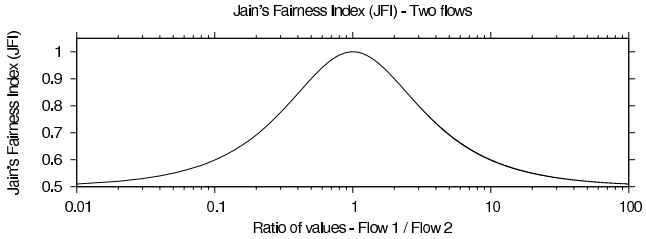


Figure 1. The range of JFI for two flows, Flow 2 = 100 (no units)

III. Resource usage and relative capability

An implicit assumption in JFI is that all of the processes being measured are equally capable of consuming the resource for which they are competing, and this is indeed the general assumption made in previous work [9], [15], including our own [1], [16]. However, when examining network flows, this is not necessarily true: some protocols may attain better performance than others given the same network conditions. It is thus necessary to take into account the flows' actual *capabilities*, in terms of the resources that it is *possible* for a flow to consume. That is, when making direct comparisons between resource consumption, JFI does not take into account the *relative capability* of the processes that are being evaluated,

and so biases may result. So, we propose a different metric when considering fairness, one that is designed to be simple but allows:

- weights to be applied that reflect relative capability, given specific resource provisioning.
- comparative assessments to be made, based on relative capability.

Further, we choose to reflect the following characteristics in the output of our metric, in comparison to JFI:

- to be able to make comparisons of fairness on a *per-flow* basis, not just a *system-wide* basis.
- to enable an assessment of fairness over time (as well as a summary statistic), allowing observation of *per-flow* and *system-wide* dynamics.

A summary of the important definitions is given in Table I.

Defn	Name	Eqn
$U_n(t)$	Normalised Resource Usage (NRU)	3
$R_n(t)$	Resource Share Ratio (RSR)	4
U_n	flow NRU (mean of the set of values $\{U_n(t)\}$)	5

Defn	Name	Eqn	Defn	Name	Eqn
U_S	NRU tuple	6	S_n	flow stability	9
U_{N+}	fair NRU	7	S_N	system stability	10
U_{N-}	unfair NRU	8	T_U	convergence time	11

Table I
SUMMARY OF IMPORTANT DEFINITIONS

A. Normalised Resource Usage (NRU)

In order to provide a richer view of the fairness information, we introduce a metric which is based on the ratio of resource usage of an individual flow with respect to its expected capability: the *Normalised Resource Usage (NRU)*. The NRU metric, $U_n(t)$, for a flow n with throughput $r_n(t)$ at time t is defined in terms of the *Resource Share Ratio (RSR)*, $R_n(t)$:

$$U_n(t) = 10\log_{10}(R_n(t)) \quad (3)$$
$$R_n(t) = w_n(t)r_n(t) \quad (4)$$

where w_n is a weight which reflects the relative capability of the flow under the conditions being examined. Key to this metric is the evaluation of $w_n(t)$, which we address in due course (Section V-A). The use of the $10\log_{10}()$ deciBel term is for convenience of representing large and small values. When $U_n(t) = 0$, then flow n is receiving a fair share of the available resource. When $U_n(t) < 0$, then flow n is receiving less than its fair share of the resource. When $U_n(t) > 0$, flow n is receiving more than its fair share. This makes it easy to make relative comparisons of fairness between flows: if any flows have $U(t) < 0$, then they are receiving less than their fair share of the resource, compared to their expected resource usage, regardless of the performance of other flows.

For a flow n with a set of values $U_n(t)$, over a given time-period, we can generate a summary by taking the mean, U_n, of the values in $U_n(t)$. We call this the *flow NRU*:

$$U_n = \overline{\{U_n(t)\}} \quad (5)$$

B. System-wide summaries

While time-based data sets let us view detailed dynamics, system-wide summaries are also important to allow comparative analyses to be made. We use our definition of $U_n(t)$ to generate summaries as follows.

Assuming a system has N flows, $1 \leq n \leq N$, a system-wide summary could be obtained by taking the mean, U_N, of U_n for all N flows. However, such a mean could hide unfairness, as positive and negative values of U_n would cancel out. So, we produce the *system NRU tuple*:

$$U_S = \langle U_{N+}, U_{N-}, S_N \rangle \quad (6)$$
$$U_{N+} = \overline{\{U_{n+}\}}, \{U_{n+}\} \subset \{U_n\} \forall U_n \geq 0 \quad (7)$$
$$U_{N-} = \overline{\{U_{n-}\}}, \{U_{n-}\} \subset \{U_n\} \forall U_n < 0 \quad (8)$$

where U_{N+} is the mean of the fair (zero) or better (positive) flow NRU values, and is called the *fair system NRU*; U_{N-} is the mean of all the unfair (negative) flow NRU values, and is called the *unfair system NRU*; and S_N is the *system stability* of all the flow NRU values, as explained in Section III-C, below.

	U_{N+}	U_{N-}	Case	Comment
unfairness	-	-ve	A	all flows unfair
	+ve	-ve	B	some flows fair or better, some flows unfair
	0	-ve	C	some flows fair, some flows unfair
fairness	+ve	-	D	all flows fair or better
	0	-	E	all flows fair

Table II
INTERPRETATION OF SYSTEM NRU VALUES

To explain the use of U_{N+} and U_{N-}, we refer to Table II. The combination of values ("$-$" denotes no value) can be grouped into those combinations that indicate a fair system and those that indicate unfairness. In the "Comment" column, 'fair' and 'unfair' are compared to the performance of the flows when the individual flows are run in a fair system. For Case A, there are no flows that have fair treatment, so the system as a whole is unfair. For Case B, some flows get more than their fair share and some less – so something in the system and/or the behaviour of some of the flows is causing unfairness to other flows. In Case C, no flows are getting more than their fair share, but some are getting less, so something in the system-wide behaviour or the behaviour of the flows themselves, is causing the unfairness. For the remaining two cases, all flows are either performing as well as in a fair system (Case E), or some are performing better than in the fair system (Case D).[3]

JFI cannot make this kind of distinction – it only indicates whether the system as a whole is fair. Additionally, use of the deciBel units allows us to make comparative performance analyses using familiar engineering semantics, which is not possible with JFI.

[3]This latter case is included for the sake of completeness, but in practical situations, it may not occur.

C. Stability

For examining stability, we use the magnitude of the co-efficient of variance (CV) (the CV is also used in [6], [9]). For a single flow n, with flow NRU of U_n and flow standard deviation σ_n across the set of values $\{U_n(t)\}$, we define the *flow stability* as:

$$S_n = |\sigma_n / U_n| \quad (9)$$

Lower values of S_n indicate better stability. For evaluating *system-wide* stability, we define the *system stability* as the mean of the set of flow stability values:

$$S_N = \overline{\{S_n\}} \quad (10)$$

D. Convergence time to fairness

In order to allow the use of $U_n(t)$ for determining when fairness has been established, we consider both *system-wide convergence time* and *inter-flow convergence time*. We assume an epoch, t_e, at which a change in the system occurs (e.g., a new flow is introduced, or a resource change occurs along the path). For the set of values $\{U_n(t)\}$, we define the *flow convergence time* to fairness T_U as:

$$T_{Un} = t_{Un} - t_e \quad (11)$$

$$\{U_n(t = t_{Un})\} \geq U_c \quad (12)$$

For our experiments we chose $U_c = -0.46$ (equivalent to an RSR value of 0.90), representing a maximum unfairness of 10% for the flow. That is, the time taken, from some epoch, for the flow to come to a point where it is within 10% of its expected throughput. Of course, the user is free to chose other values of U_c as appropriate.

For a subset of M flows, with a corresponding set of NRU for flows $U_M = \{\{U_1(t)\}, ...\{U_M(t)\}$ (for $M < N$), we define the *inter-flow convergence time* to fairness by replacing $\{U_n(t)\}$ in Equation 12 with $\{U_m(t)\}$. For the *system convergence time* to fairness, we set $M = N$. We leave the choice of t_e, the subset of flows, and the value of U_c to the user, based on his/her own context and requirements.

IV. Methodology

Note that while our intention is to demonstrate our new metric, we feel that it is important to use a testbed and real protocols to provide confidence that our metrics can be used in practice. So, we use data from our ongoing work [2].

We take a practical approach, generating data flows using a version of the tool *iperf*[4], modified to report additional TCP state.[5] We transmitted flows over a simple testbed, sending two flows over a single bottleneck link. This allows observations of the end-to-end behaviour of the flows over the bottleneck link, and measurement of their relative performance. We evaluated how *fair* the resource share was for the two flows by using the end-to-end throughput achieved by each flow, as reported at 1s intervals by *iperf*.

To demonstrate our approach, we choose three transport protocols: TCP NewReno, CUBIC [18] and CCID2 [8] of the Datagram Congestion Control Protocol [13]. Our choice is arbitrary – we have previous experience of using these particular protocols for measurement-based analysis, and so we are familiar with their behaviour. TCP NewReno is the "standard" TCP; CUBIC is currently the default version of TCP on Linux; CCID2 is on the IETF Standards Track and an implementation is available in the Linux kernel.

A. Testbed

Our testbed set-up was the well-known dumbbell arrangement as depicted in Figure 2 and used in previous similar studies [9], [15]. This simple testbed helps to reduce the factors of error or unknown behaviour that may affect the results and concentrate on the protocol behaviour. As noted in [7], "Simple topologies, like a 'dumbbell' topology with one congested link, are sufficient to study many traffic properties." Full details of the testbed are given in [2].

The testbed consisted of two senders, two receivers and a router to provide the network delay and bottleneck. All network connections were 100Mb/s full-duplex Ethernet. Our measurement runs consisted of generating two flows, using *iperf*, for a pair-wise comparison: Flow 1 was from Sender 1 to Receiver 1, and Flow 2 was from Sender 2 to Receiver 2. The duration of each measurement run was 300s, with Flow 1 starting at 0s, and Flow 2 starting at 30s to avoid initial synchronisation effects. After some calibration tests, we performed 25 measurement runs for each of the TCP protocols running against CCID2 at each of the following RTT values: 25ms, 50ms, 75ms, 100ms, 125ms, 150ms, 175ms, 200ms. The choice of CCID2 as the protocol against which comparisons were made was arbitrary but sufficient for our needs; we could have chosen to compare against either of the other two protocols, or conducted a complete set of pair-wise experiments across all the protocols.

The senders and receivers ran Linux kernel version 2.6.22.6, and we used "out-of-the-box" configuration for the end systems[6], rather than tuning the stack for high-speed operation (as in [1], [9], [15]), in order to gauge the performance under (arguably) the most likely configuration of the end-systems.

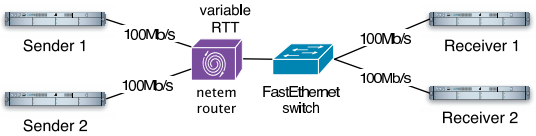


Figure 2. Testbed configuration

We configured *iperf* to report throughput values at 1s intervals, and evaluated $J(t)$ (Equation 2) and $U_n(t)$ (Equation 3) for each throughput measurement. Flow 2 started at $t = 30s$,

[4] http://dast.nlanr.net/Projects/Iperf/

[5] This modified version of *iperf* is available from the authors.

[6] CCID2 was configured as recommended in http://www.linux-foundation.org/en/Net:DCCP.

so we used values from $t = 60s$ (to permit flows to stabilise) to $t = 300$ (the end of the measurement run) in order to calculate values for $J(t)$, $U(t)$, and other metrics.

V. Intra-protocol behaviour

In this section, we describe the behaviour of two instances of the same flow: as well as showing typical behaviour, this allows us to establish the weighting, $w_n(t)$ to be used in Equation 4. Note that we have used two flows in order to present a simple and clear discussion, but our methodology can be applied to more than two flows. We performed experimental runs of the protocols at each of the RTT values listed earlier. We start Flow 1 at $t = 0s$ and Flow 2 (of the same protocol) at $t = 30s$, and then make our evaluation from $t = 60s$. Table III shows the mean values recorded over 5 runs.

	RTT (ms)							
	25		50		75		100	
Flow	1	2	1	2	1	2	1	2
NewReno	45	49	42	51	42	52	43	51
CUBIC	42	52^B	51	43	47	45	44	48^D
CCID2	43^A	51	46	47	44	50	45^C	48

	RTT (ms)							
	125		150		175		200	
Flow	1	2	1	2	1	2	1	2
NewReno	47	47	47	47	47	46	43	43
CUBIC	45	46	42	46	46	41	41	41
CCID2	39	40	36	38	30	30	33	32

Table III
Throughput values (Mbits/s) for two flows of the same protocol (mean over 5 runs, from $t = 60s$ to $t = 300s$)

A. Weights for NRU

The self-fairness tests form an important distinguishing feature in our methodology and use of our new fairness metric: they are used to assign weights (Equation (4)) that reflect each flow's capability to consume the available resource. From Equation (4), we define:

$$w_n(t) = 1/_R r_n(t) \quad (13)$$

$$w_n = 1/\overline{_R r_n} \quad (14)$$

where ${}_R r_n(t)$ is the expected throughput of that flow under the conditions being examined. Equation (13) is a general expression, and we simplify this for our needs by using Equation (14), which represents the weight as a scaling factor, evaluated from the mean throughput. In our case, the value of w_n for each flow is taken from the mean of the two values (Flows 1 and 2) in Table III, at each RTT value.

VI. Inter-protocol behaviour

We now examine the use of Normalised Resource Usage. We look at the interaction of our chosen protocols by examining the behaviour of two different flows across the testbed. Flow 1 was started at 0s and was a CCID2 flow. Flow 2 was started at 30s and was NewReno or CUBIC. We executed 25 runs of each pair-wise experiment for each of the RTT values as explained in Section IV-A. We recorded the end-to-end throughput reported by *iperf* at 1s intervals from $t = 60s$ (allowing 30s for Flow 2 to stabilise) to $t = 300s$ (the end of the experimental run). These throughput values were used with Equation (1). We also noted the mean throughput of the two flows over the 25 runs.

A. Summary of observed behaviour

To summarise the behaviour, the values of JFI generated using Equation (1) are given in Table V (the mean JFI values and CV over 25 runs) and shown in Figure 3. Table IV gives the mean throughputs over 25 runs for each protocol, where Flow 1 in the table is always CCID2 and Flow 2 is the other flow (NewReno or CUBIC). We note that the fairness of CCID2 and NewReno is good across the range of RTT values examined, and this is encouraging as it meets a goal of CCID2 to be "TCP-like". However, we observe that CUBIC has poor fairness with CCID2 beyond an RTT value of 25ms, with the TCP variant using more capacity than CCID2.

	RTT (ms)							
	25		50		75		100	
Flow	1	2	1	2	1	2	1	2
NewReno	56	38	55	39	55	39	52	41
CUBIC	39^E	55^F	26	67	18	74	17^G	73^H

	RTT (ms)							
	125		150		175		200	
Flow	1	2	1	2	1	2	1	2
NewReno	43	44	38	37	33	32	30	29
CUBIC	18	73	23	70	22	71	22	70

Table IV
Throughput (Mbits/s) for pair-wise tests against Flow 1 = CCID2 (mean over 25 runs, from $t = 60s$ to $t = 300s$)

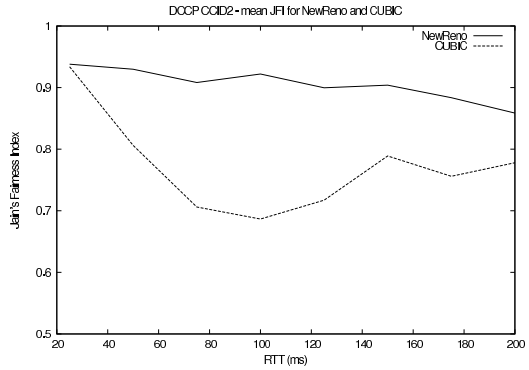


Figure 3. JFI summary of pair-wise inter-protocol behaviour (See Table V)

	RTT (ms)			
	25	50	75	100
Reno	0.94, 0.01	0.93, 0.02	0.91, 0.03	0.92, 0.03
CUBIC	0.93^I, 0.01	0.80, 0.09	0.71, 0.14	0.69^J, 0.15

	RTT (ms)			
	125	150	175	200
Reno	0.90, 0.05	0.90, 0.05	0.88, 0.05	0.86, 0.09
CUBIC	0.72, 0.14	0.79, 0.11	0.76, 0.13	0.78, 0.13

Table V
⟨JFI, CV⟩ values for pair-wise tests against CCID2 (mean over 25 runs, from $t = 60s$ to $t = 300s$) (See Figure 3)

This behaviour is to be expected: as the RTT (and so the BDP) increases, from Table III, it can be seen that, although flows become poorer at consuming capacity at higher RTT values, CUBIC and NewReno achieve higher throughput than CCID2 as RTT increases. So the unfairness observed in Figure 3 may not necessarily be a concern. Note, however, that as the RTT increases beyond ∼125ms, the fairness improves as CCID2 throughput increases.

In the case of CUBIC, we see from Figure 3 and Table V that JFI ≤ 0.8 at all values of RTT above 25ms, i.e. there would appear, at first sight, to be varying degrees of unfairness between CUBIC and CCID2. In Table IV, we see that CUBIC always has a higher throughput than CCID2.

B. Assessing relative capability

It is not sufficient to consider only the JFI of the pair-wise tests in order to make a true assessment of fairness. We must also consider that CUBIC is designed for high BDP paths, whereas CCID2 is designed to exhibit "TCP-like" behaviour. So, we must take into account the likely throughputs under circumstances where they may be competing equally with the other flow in the experimental run. *Is the higher throughput of the CUBIC flows due simply to that protocol using the capacity that CCID2 is not able to use effectively at higher BDPs?*

We can answer this question by comparing the mean throughput figures in Table IV (which records the mean throughputs of the *pair-wise* experiments) and Table III (which records the mean throughputs of *two flows* of the same protocol). So, we are trying to assess fairness not just by looking at the JFI values, but also by comparing the protocol performance of CCID2 against a protocol with which it has very fair behaviour at all RTTs, i.e. itself.

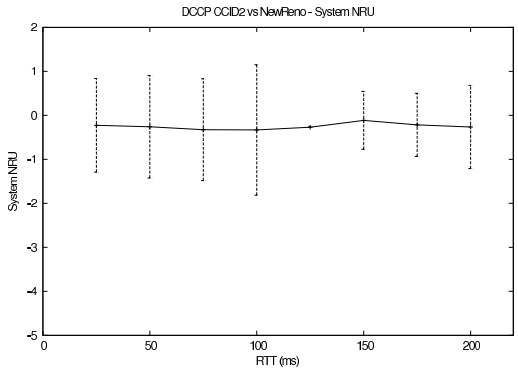
We first consider CUBIC at RTT=25ms. Recall that in our pair-wise experiments, Flow 1 is always CCID2. So we compare the Flow 1 values in Table III for CCID2, and the values at the same RTT in Table IV. We find that at RTT=25ms, the throughputs of both CCID2 and CUBIC are similar: CCID2 (the cell in Table III marked ^{A}) is 43Mb/s, and CUBIC (the cell in Table III marked ^{B}) is 52Mb/s; CCID2 (the cell in Table IV marked ^{E}) is 39Mb/s, and CUBIC (the cell in Table IV marked ^{F}) is 55Mb/s. So, there is very good fairness. The JFI value is 0.93 (the cell in Table V marked ^{I}), however it does not really reflect the relative performance of each flow.

For CUBIC at RTT=100ms, we find that the throughputs of both CCID2 and CUBIC are very different: CCID2 (the cell in Table III marked ^{C}) is 45Mb/s, and CUBIC (the cell in Table III marked ^{D}) is 48Mb/s; CCID2 (the cell in Table IV marked ^{G}) is 17Mb/s, and CUBIC (the cell in Table IV marked ^{H}) is 73Mb/s. It seems clear that in this case, the "non-TCP-like" behaviour of CUBIC has a detrimental effect on the end-to-end throughput of CCID2: CUBIC is being more aggressive than CCID2 and causing unfairness. The JFI value of 0.69 (the cell in Table V marked ^{J}) does show some unfairness, but it does not make clear the relative magnitude of unfairness: the CCID2 throughput is much less than half of what it would be against another CCID2 flow.

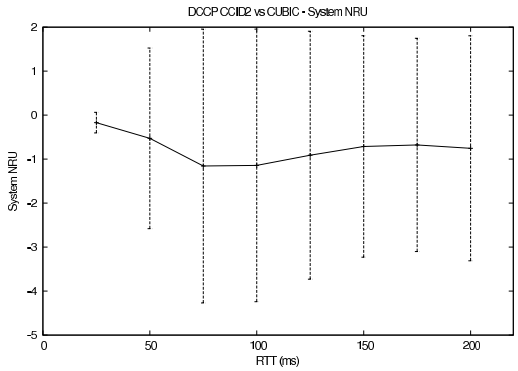
C. NRU analysis

For NRU, we use weights w_n generated as explained in Section V-A. Of course, plotting the set of values $\{U_n(t)\}$ would show details of the performance for each flow, but in order to make a comparison with JFI, we choose to show system-wide summaries based on the NRU.

For each pair-wise test, we produce NRU tuples in Table VI, with values plotted in Figure 4. Note with only 2 flows, the S_N value is not meaningful, and so we plot the tuple $\langle U_{N+}, U_N, U_{N-}\rangle$, where U_N is the mean of the values in the set $\{U_n\}$. For a larger number of flows, we may choose other visualisations, as appropriate, e.g., to plot the tuple $\langle +S_N/2, U_N, -S_N/2\rangle$, or to use a box-and-whisker plot with the set of flow NRU values.



(a) NewReno vs CCID2



(b) CUBIC vs CCID2

Figure 4. System NRU of pair-wise inter-protocol behaviour (See Table VI)

Considering the indicators in Table II, we see that most of the NRU tuples in Table VI are of Case B (to 1dp, the NRU tuple for CUBIC at RTT=25ms is Case C). However, at $t = 125ms$ for NewReno, there were no positive NRU values, so $U_N = U_{N-}$, which gives Case A. The JFI plot in Figure 3 cannot give such information.

Note that because we have only two flows, we can see instantly the range of performance values. For example, taking the case for CUBIC at RTT=100ms, in Table VI, we see that $U_{N-} = -4.24$, so clearly one of the flows has less than half the capacity it would expect, while $U_{N+} = 2.00$ shows that one of the flows has much greater capacity than it would expect.

The use of the deciBel notation for NRU allows easy inspection of relative performance from the NRU graphs. If

	RTT (ms)			
	25	50	75	100
NewReno	$\langle 0.84, -1.29, 6.66 \rangle$	$\langle 0.91, -1.43, 6.35 \rangle$	$\langle 0.83, -1.48, 5.01 \rangle$	$\langle 1.15, -1.81, 6.31 \rangle$
CUBIC	$\langle 0.06, -0.40, 1.93 \rangle$	$\langle 1.53, -2.58, 5.50 \rangle$	$\langle 2.00, -4.27, 3.80 \rangle$	$\langle 2.00, -4.24, 3.83 \rangle$
	RTT (ms)			
	125	150	175	200
NewReno	$\langle -, -0.27, 1.23 \rangle$	$\langle 0.54, -0.77, 8.13 \rangle$	$\langle 0.50, -0.93, 4.69 \rangle$	$\langle 0.68, -1.21, 5.06 \rangle$
CUBIC	$\langle 1.90, -3.73, 4.37 \rangle$	$\langle 1.80, -3.23, 5.00 \rangle$	$\langle 1.75, -3.10, 5.06 \rangle$	$\langle 1.80, -3.31, 4.79 \rangle$

Table VI
NRU TUPLES FOR PAIR-WISE TESTS AGAINST CCID2 (MEAN OVER 25 RUNS, FROM $t = 60s$ TO $t = 300s$ AND $t_e = 60s$) (SEE FIGURE 4)

we compare the value of U_N (the middle line) in each graph of Figure 4 with the corresponding lines for each of the two protocols in Figure 3, we see that, as a system-wide mean, the NRU shows better fairness between CCID2 and the other flows than seen using JFI, reflecting better the relative capability of each protocol, and the consistency of the behaviour in the flatter line of each graph in Figure 4 compared to Figure 3. The NRU tuple and the NRU plot allow us to see clearly the range of NRU values (the top and bottom of the vertical bars).

D. Convergence times

Table VII shows convergence times. The epoch, t_e, was taken as $t = 30s$, i.e. the time that the second flow was introduced. We see here that while NewReno is more likely to converge to fairness than CUBIC, the latter appears to do so more quickly than NewReno. However, we also know that CUBIC shows greater unfairness to CCID2 than NewReno, and so the convergence is likely to be short-lived (which is confirmed on visual inspection of the plotted data for individual flows, and is not presented here).

	RTT (ms)							
	25	50	75	100	125	150	175	200
NewReno	109	208	-	157	121	62	198	172
CUBIC	71	157	61	84	-	-	61	62

Table VII
CONVERGENCE TIME (T_U) (SECONDS) VALUES FOR PAIR-WISE TESTS (MEAN OVER 25 RUNS, FROM $t = 60s$ TO $t = 300s$)

E. Larger numbers of flows

We now show the use of the NRU with larger numbers of flows. Note that these simulation results should not be considered as rigorous as those for our pair-wise tests above: our focus is to show the characteristics of the NRU and the actual protocols used are not important.

We simulated a 14-flow system using ns2 patched with the ns2 Linux extension from [17], allowing Linux kernel code for different transport protocols to be executed from within ns2. We used 14 different TCP variants, one of each type supported by [17]. We used the same scenario and topology as for our testbed albeit with 14 senders and 14 receivers. Flows were started at 30s intervals (from $t = 0$) until all flows were active. The simulation lasted for 900s and we analysed data points from $t = 420s$ to $t = 900s$ in order to avoid any start-up artefacts from the flows. Table IX, Table VIII and Figure 5 shows the JFI and system NRU values (plotted as the tuple $\langle U_{N+}, U_N, U_{N-} \rangle$) for the 14 variants. The NRU weights were calculated in a similar fashion to that for our testbed, i.e. using a run of 14 flows of the same type. We conducted only a single run of the simulation in each case.

	RTT (ms)							
	25	50	75	100	125	150	175	200
JFI	0.51	0.30	0.25	0.24	0.22	0.20	0.20	0.19

Table VIII
JFI VALUES FOR 14 FLOWS (SINGLE RUN) (SEE FIGURE 5(A))

When we compare the JFI graph and NRU graph in Figure 5, we see that the JFI shows the system becoming increasingly unfair as the RTT increases, but the NRU shows that the system fairness improves after 100ms, as we take into account the relative capability of each protocol flow at those higher BDPs. Also, JFI depicts unfairness but masks the relative magnitude of reduction in performance, an aspect which is clearly visible in the NRU values (Table IX). Again, with reference to Table II, we have NRU tuples that are of both Case A and of Case B (to 1dp, the tuple at $t = 125ms$ is Case C), a distinction that JFI cannot make.

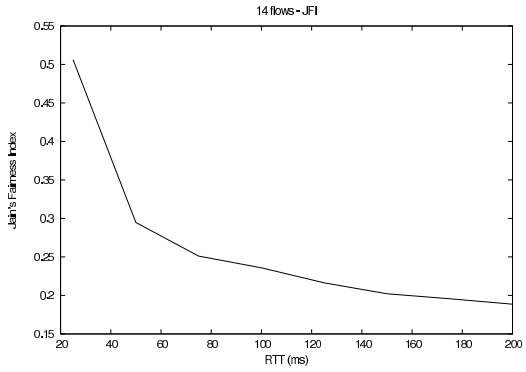
VII. CONCLUSION

We have shown a simple approach to making richer assessments of inter-flow *fairness* using end-to-end measurements of flow throughput. Compared to the the use of Jain's Fairness Index (JFI), our approach, which defines the Normalised Resource Usage (NRU) for a flow, uses weights for each flow which reflect its relative capability. NRU tuples can be used to show the range of NRU values, which can indicate whether flows are performing better or worse than they might on the same network path but with like flows. NRU allows assessment of per-flow fairness as well as system-wide fairness. Apart from the evaluation of the weights, the cost in measurement is the same as would be if using JFI. Indeed, it would be possible to evaluate the NRU retrospectively if data were already available from previous studies and a suitable testbed could be found to generate the weights. We also define a use of the NRU to assess system *stability* in terms of the Coefficient of Variance of the flow data, and *convergence times* for flows.

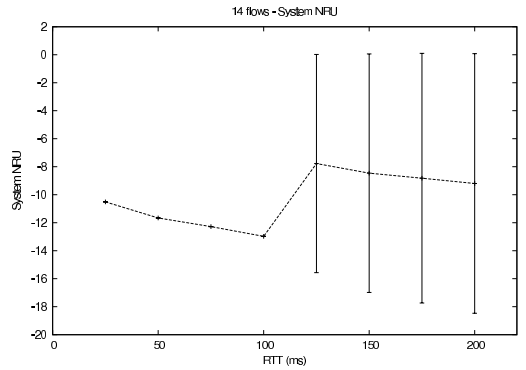
We demonstrated the use of the metric with a rigourous testbed study of three protocols (TCP NewReno, CUBIC and DCCP/CCID2), and a multi-flow simulation. We compared a NRU-based analysis with an analysis based on the use of JFI, showing the richer information provided by the NRU. Although we chose to use end-to-end throughput for our evaluation, the NRU could be used with other flow performance measures, such as end-to-end delay for a data transfer, or cost.

RTT (ms)			
25	50	75	100
$\langle -, -10.51, 3.21 \rangle$	$\langle -, -11.70, 1.94 \rangle$	$\langle -, -12.28, 1.80 \rangle$	$\langle -, -12.97, 1.68 \rangle$
RTT (ms)			
125	150	175	200
$\langle 0.02, -15.57, 2.56 \rangle$	$\langle 0.10, -16.98, 1.57 \rangle$	$\langle 0.10, -17.74, 2.88 \rangle$	$\langle 0.10, -18.47, 2.44 \rangle$

Table IX
SYSTEM NRU TUPLES FOR 14 FLOW TEST IN NS2 (SEE FIGURE 5)



(a) JFI (See Table VIII)



(b) System NRU (See Table IX)

Figure 5. System NRU of 14-flow inter-protocol behaviour

A. Limitations

The NRU's key benefit, the use of weights that provide the assessment of relative capability, is potentially also a limitation of the method. The key question is "How do we find appropriate values for the weights?" Our approach, with two flows and 14 flows, has used calibration runs in order to assess what each flow would achieve if competing with itself. Unfortunately, the use of calibration runs does not scale well as the number of flows increases. However, it should be possible to use other techniques to estimate the weights as the need arises, and the exact nature of the approximation is likely to be dependent on the system set-up. Meanwhile, it is to be noted that most studies in the literature typically make experimental assessments with small numbers of flows, e.g. two flows (as we have done), and so for such purposes, our methodology is quite suitable and provides more accurate evaluation of performance than JFI.

A further limitation is that it is known that the Coefficient of Variance (CV), used for S_n, becomes highly sensitive with small values. So, for small values, S_n may give an inaccurate measure of instability.

REFERENCES

[1] M. Bateman, S. Bhatti, G. Bigwood, D. Rehunathan, C. Allison, T. Henderson, and D. Miras. A Comparison of TCP Behaviour at High Speeds Using ns-2 and Linux. In *CNS2008 - 11th Communications and Networking Simulation Symposium*, April 2008.

[2] S. Bhatti, M. Bateman, D. Rehunathan, T. Henderson, G. Bigwood, and D. Miras. A Comparative Performance Evaluation of DCCP. In *SPECTS2008 - 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, June 2008.

[3] B. Briscoe. Flow rate fairness: dismantling a religion. *SIGCOMM Computer Communication Review*, 37(2):63–74, 2007.

[4] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14, 1989.

[5] S. Floyd. Metrics for the Evaluation of Congestion Control Mechanisms. RFC 5166 (Informational), Mar. 2008.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *Proc SIGCOMM 2000*, pages 43–56, 2000.

[7] S. Floyd and E. Kohler. Internet research needs better models. In *Proceedings of HotNets–I*, October 2002.

[8] S. Floyd and E. Kohler. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. RFC 4341 (Proposed Standard), Mar. 2006.

[9] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu. A Step Toward Realistic Performance Evaluation of High-Speed TCP Variants. In *Fourth International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet06)*, Feb 2006.

[10] V. Jacobson and M. J. Karels. Congestion Avoidance and Control. *Proc SIGCOMM 1988*, pages 314–329, September 1988.

[11] J. Jaffe. Bottleneck Flow Control. *IEEE Transactions on Communications*, 29(7):954–962, July 1981.

[12] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, volume 49, pages 237–252, 1998.

[13] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340 (Proposed Standard), Mar. 2006.

[14] S. Kunniyur and R. Srikant. End-to-end Congestion Control Schemes: Utility Functions, Random Losses and ECN Marks. *IEEE/ACM Transactions on Networking*, 11(5):689–702, October 2003.

[15] Y.-T. Li, D. Leith, and R. N. Shorten. Experimental Evaluation of TCP Protocols for High-Speed Networks. *IEEE/ACM Transactions on Networking*, 15(5):1109–1122, Oct 2007.

[16] D. Miras, M. Bateman, and S. Bhatti. Fairness of High-Speed TCP Stacks. In *Proc. AINA2008 - IEEE 22nd International Conference on Advanced Information Networking and Applications*, March 2008.

[17] D. X. Wei and P. Cao. NS-2 TCP-Linux: an NS-2 TCP implementation with congestion control algorithms from Linux. In *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*, page 9, New York, NY, USA, 2006. ACM Press.

[18] L. Xu and I. Rhee. CUBIC: A new TCP-Friendly high-speed TCP variant. In *Third International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet05)*, Feb 2005.