Internet—Fairer is Faster

More considerate Internet usage would be faster

By Bob Briscoe

The Internet is founded on a simple insight: a shared communications channel is more efficient than a dedicated one that sits idle much of the time. We share neighbourhood links on broadband networks. Then we share again—the mesh of links in national networks and the backbone cables criss-crossing the globe. That's why you'll rarely get 2Mbps from an Internet line sold as "up to 2Mbps". Even a 40Gbps backbone cable can't give more than 1Mbps each if you share it equally with 40,000 others—however big your access line. But, even though you don't get your full 2Mbps, the great thing about the Internet is you get *a* share of everything. You and a thousand million computers and devices can share any of the equipment in the whole public Internet, all without asking.

But how big a share do you get? The classic sharing algorithm is the transmission control protocol, the sister to the Internet protocol. They are often spoken of together as “TCP/IP”—the controller and the engine of the Internet. Once data flows fill any link to capacity, TCP will try to share the bit rate equally.

This seems fair enough. It served the Internet well in the early days—indeed, it has become the definition of Internet fairness. However, unfortunately, it merely gives an illusion of fairness. You can keep to TCP's rule, but still get as big a share as you want. If you keep coming back for more, you keep getting equal shares with someone who only uses capacity sparingly. And you get multiple shares if you run multiple data flows at once.

It’s as if scientists working on a food rationing system have convinced themselves that it will be fair if they make all the ration boxes the same size, irrespective of how many boxes each person takes at a time or how often each comes back for more.

On top of this first delusion, there's a second. TCP doesn't control the sharing anyway, at least not where sharing matters most. Network operators override TCP to share out the capacity of the most contended parts of the Internet— the shared neighbourhood links.

- The tensions between the two approaches are tearing the Internet apart
- A new approach can force everyone to be considerate to everyone else
- Paradoxically, everyone will get much more out of the Internet, just by sharing it out more considerately

You might be shocked to discover that, by design, your share of the capacity of any transmission line in the Internet is determined by how polite your software chooses to be. Within your own computer's operating system, whether Windows, MacOS or Linux, there's TCP software that politely limits the share of the Internet it uses. And most programs you run choose to use TCP to access the Internet, even though they don't have to. Currently, about 90% of the Terabytes on the Internet at any one time are under the control of TCP's voluntary politeness.

The Internet was designed so that network operators played no mediating role in the conflicting demands of the thousand million personal computers, mobile devices and servers on the Internet. Yet it all seems to work; an amazing global outpouring of self-denial, like the “after you” protocol we use when people approach a door at the same time—but billions of after-yous each day, between complete strangers, fierce competitors, and even mortal enemies.

This bravely naïve way to control congestion was retrofitted comparatively late in the Internet’s 35-year history. By Oct 1986 the Internet started to suffer a series of “congestion collapses.” Traffic overran the available capacity. The network software of the day continued to try to retransmit, causing everyone’s useful throughput to plummet for hours on end.

There were many attempts to solve the problem by prioritisation in router queues, but by mid-1987, Van Jacobson, then a researcher at Lawrence Berkeley Laboratory, had coded a set of elegant distributed algorithms in a patch to TCP.

Jacobson’s congestion control accorded well with the textbook design principle of the Internet. All traffic control is consigned to the computers around the edges of the Internet (using TCP), while network equipment only routes and forwards packets of data (using IP). System administrators quickly added Jacobson’s patch to the thirty thousand or so computers that constituted the Internet at that time. It solved the problem because it was more polite than the overly aggressive code it replaced. No-one sneakily held on to their old aggressive version to exploit everyone else’s new-found self-denial, because the patch did such an astonishing job of improving TCP's general performance as well.

And still today, almost every Internet application will delegate all its traffic control functions to the TCP code in the operating system. So at any moment, billions of instances of Jacobson’s congestion control

algorithms are interacting to determine the bit rate we each get as our bits squeeze through one bottleneck or another as they traverse the gigantic mesh of links that forms the Internet.

TCP works by constantly increasing its rate until some link along the way can't handle the data flow, and has to discard some packets. The sending computer hears about the resulting gap in the data stream from the receiver and politely halves its rate as it retransmits the missing data. The constant increase then continues from this new low (**Figure 1**). All told, every TCP algorithm continually alters how Internet capacity is shared, by making perhaps ten to a hundred rate adjustments every second—multiplied a billion-fold around the Internet.

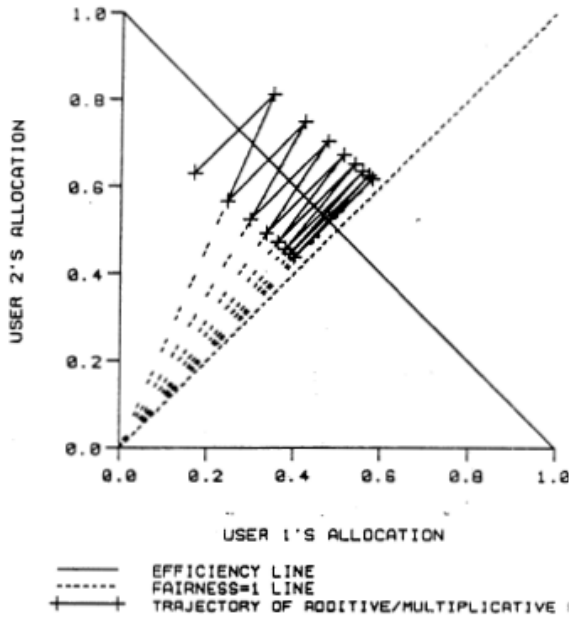

Figure 1 Dah Ming Chiu's original Apr 1987 viewgraph proposing how TCP could seek out the sweet spot in the middle that the zig-zag trace is converging towards, where the two users get shares that are both 'fair' and efficient.

A massively distributed algorithm that has survived two decades of Internet growth despite relying on self-denial has an entrancing allure, particularly to academics. Jacobson's paper on TCP's congestion avoidance is the fifth most cited in all of computer science. Some 5000 other papers have discussed fair resource allocation in relation to TCP.

The combination of academic endorsement and near-universal usage has gradually elevated TCP's way of sharing capacity to the moral high ground. Indeed, it has altered the very language network engineers use.

From the beginning, equal rates were not just "equal," they were "fair." And in recent years the cosy-sounding concept of "TCP friendliness" has been coined as a goal for the growing range of new congestion control algorithms designed to extend the Internet to meet modern requirements, such as smoother rate variation for streaming media or faster start-up for high speed links. By definition, a TCP-friendly algorithm takes about the same share of bit-rate as TCP would, even though it might exhibit very different dynamics. Nowadays, if a new algorithm isn't TCP-friendly, it is considered suspect and will not get through the Internet standards process.

Sadly, an equal bit-rate for each data flow is likely to be extremely *unfair*, by any realistic definition of fairness. The problem is that the real world has two extra degrees of freedom beyond those recognised in TCP's fair-sharing rule. So everyone can comply with the letter of TCP's rule, but still just take however much they want.

TCP's first overlooked dimension is the passage of time itself. As in the food rationing analogy earlier, fair allocations depend on how often an individual keeps coming back for more. So judging fairness at each moment in isolation from all previous moments leads to problems.

Consider a typical network situation: a neighbourhood of 100 broadband customers of an Internet service provider. Each has a 2 megabit-per-second access line connected to a single, shared 10 Mb/s regional link (Figure 2). The provider can get away with

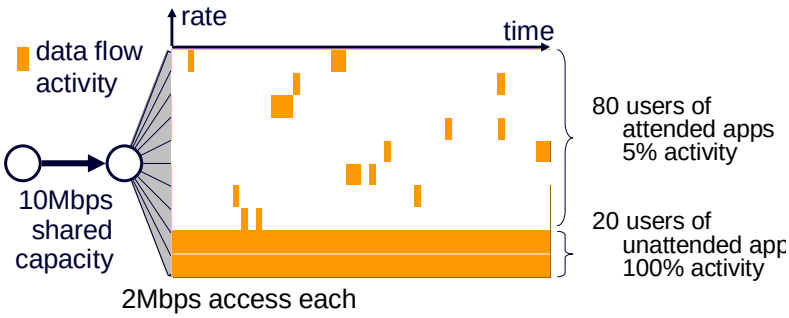

Figure 2 TCP's First Missing Dimension: Activity over time. The TCP algorithms of all 100 users take shares of the ISP's 10Mbps link. But those active at any one moment take equal rates, regardless of how active each user is.

such a small shared link because most of the customers (say 80 of the 100) don't use it continuously, even over the peak period. They might think they are busily clicking at their browsers all the time and getting new messages, but this only translates into download activity for perhaps 5 percent of the time each. However, the other twenty are heavy users, who download continuously, perhaps using file-sharing programmes that run unattended.

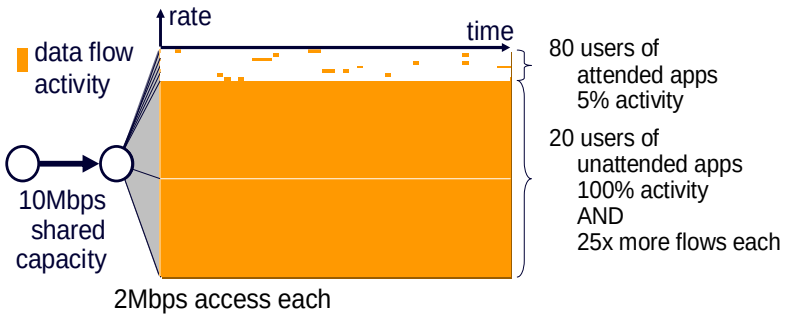

Figure 3 TCP's Second Missing Dimension: Number of Instances of the TCP Algorithm.
The TCP algorithms of all 100 users take shares of the ISP's 10Mbps link. But an application developer can take 25 times more shares by just running the TCP algorithm 25 times in parallel.

Compounded with 20 times higher activity over time, the unattended apps shown use the shared link with 500 (=20x25) times greater intensity.

In other words, at any one moment, data is flowing to about 24 users—all 20 heavy users, and 4 of the 80 light ones. TCP gives 20 shares of the bottleneck capacity to the heavy users and only 4 to the light users. But, at the next moment, the 4 light users will have stepped aside and another 4 will take over their shares. However, the 20 heavy users will still be there to claim their next 20 shares. That's not fair to the 80 light users.

That's not all. TCP overlooks a second degree of freedom: a programmer only has to run the TCP algorithm multiple times in parallel to get multiple shares (Figure 3), much like a food rationing system that allows duplicate ration coupons. One capacity share is taken by each *instance* of the TCP algorithm. The potential for software applications to play this second trick has always been recognised by Internet scientists and engineers. For example, the early Web browsers opened four TCP data flows in parallel.

Indeed, it would have been remarkable if software had *not* evolved to exploit this degree of freedom further. Nor did it take long. Parallel incremental downloading (or "swarming") first appeared on the Internet in 1999. By 2001 it had become routine, built into the protocols like BitTorrent that are used by many of today's peer-to-peer file-sharing applications. Each peer downloads chunks of a file from a number of other peers, usually while they in turn are still downloading other chunks from still other peers.

The networking community didn't see this as a circumvention of the TCP-friendliness rule. After all, each transfer between peers used TCP, so each data flow "correctly" got one share of any bottleneck it encountered. But using parallel connections to multiple machines was a new degree of freedom that hadn't even been thought of when the rules were written.

In Edwin A Abbott's classic 1884 novel "Flatland: A Romance of Many Dimensions" the narrator, A Square, dreams of convincing the monarch of Lineland that a second dimension exists. TCP-land suffers from a similar affliction – it is two dimensions short relative to the world the rest of us inhabit. It overlooks how much of the time TCP is running and how many other instances of the TCP algorithm a user has running at the same time.

In Flatland, A Square is overawed by Sphere, who seems able to remove shapes from Flatland at will and later teleport them into far-off places. Similarly, in TCP-land, the continuing prevalence of TCP looks like an amazing global outpouring of self-denial. But with the aid of two extra dimensions we see the more dismal reality—endemic exploitation of the overlooked dimensions.

Let's look again at the hapless 80 broadband customers who are truly active only 5 percent of the time. Were all 100 customers doing nothing more than Web browsing and e-mail, they would each get nearly the full benefit of their 2 Mb/s connection—5 customers at a time would just squeeze into the 10Mb/s shared link. But if 20 of these users started continuous downloading, the TCP algorithm would send everyone else's bit-rate plummeting—from a robust 2 Mb/s to an anaemic 20 kb/s—worse than dial-up!

One solution is to throw capacity at this problem—upgrade the 10 Mb/s pipe that everyone is sharing. And many Internet providers do just that. But it's like throwing water up hill.

Imagine two competing Internet service providers, both with this same mix of heavy and light users. One quadruples its capacity, the other doesn't. The one who upgrades increases its monthly fees to cover the cost

of leasing four times more shared capacity. But TCP still shares out the capacity in the same way, so the light users who used to have a measly 20 kb/s share now have a measly 80 kb/s—still barely better than dialup!

But these numbers can't be right. Service providers would have stopped investing in capacity if this was true, and they have not. Certainly the numbers aren't exact and it's a hugely oversimplified model. But actually the mix of demand is in the same ball-park as a typical Internet scenario. So what's going on? Continuing investment is partly explained by government subsidies, common in the Far East, and partly by weak competition, common in the United States. But even in the competitive Internet markets of Europe, investment continues. The explanation is that network operators *override* the shares of capacity that TCP would give. Where the problem is worst, on the shared links at the outermost edges of the Internet, they limit each customer's share using a "fair queuing" scheduler. And some ISPs are deploying boxes that throttle heavier usage at peak periods by discarding any packets that exceed a certain rate.

The appeal of TCP's universal sharing algorithm is actually a double delusion. First, it's is not actually sharing out capacity well because it overlooks two degrees of freedom. And second, TCP is not actually being allowed to determine capacity shares anyway, except on the links further into the Internet where there's enough excess capacity for it not to matter how TCP shares it.

So although the Internet seems to be working fairly well superficially, it's only because providers are masking the flawed capacity-sharing of TCP. They are having to unnecessarily over-provide costly capacity and they are imposing scheduling and throttling.

Ironically, there's a far better solution than throttling anyway. It would allow light browsing to be blisteringly fast as if on a campus LAN, but hardly prolong heavy downloads at all.

The solution comes in two parts. The first part is completely counter-intuitive to those who have always believed fairness means equal flow rates. It takes multiple instances of TCP as the norm. It even makes it easier for programmers to use multiple instances of TCP, by supplying a new "weighted TCP." When a programmer starts this new TCP to transfer some data, they will be able to say "behave like 12 TCPs" or "behave like 0.25 of a TCP," by setting a new 'weight' parameter. Then, whenever your data comes up against others all trying to get through the same bottleneck, you'll get twelve shares, or a quarter of a share.

The second part of the solution gives everyone good reason to use the weights sparingly. But first, let's check how this scheme ensures the blisteringly fast browsing rates I just promised.

The key is to set the weights high for the light usage and low for the heavy. So when they're competing, the light data flows can go much faster. A light flow finishes much sooner, so once it's out of the way, heavy flows can expand to a higher bit rate sooner than otherwise (Figure 4). So the heavy flows see no less capacity on average.

The weighting scheme uses the same widely used strategy as the restaurant manager who says, "Get the individual orders out right away, before coming back to help with this large business dinner."

However, today's Internet has the balance of the weights the other way round – heavy downloads have effectively set their weights high, by opening multiple instances of TCP, while the weights of light usage are lower. That brings us to the second part of the problem – how can we encourage everyone to flip the weights round the other way?

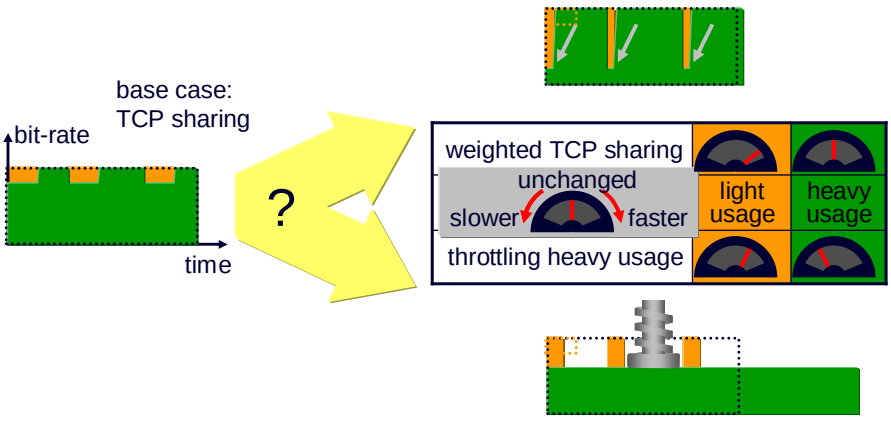

Figure 4 Throttling makes light usage a little faster by slowing down heavy users. Whereas weighted TCP sharing allows light usage to go much faster without hardly affecting heavy users.

This is one of those problems like global warming, where everyone happily pursues what's best for them – leaving lights on and driving a big car – despite the effect on everyone else – the build-up of greenhouse gases. On the Internet, it's not the volume of gigabytes you download – that doesn't matter if there's no congestion. What matters is the volume you download when everyone else wants to as well. Or, more precisely, the volume you download

weighted by the prevailing percentage level of congestion when you do. This is called your congestion volume, measured in bytes. It's the CO_2 of the Internet.

As with CO_2, the way to cut back is to set up limits. There are those who believe everyone should be free to use the Internet without any restriction at all. Quite probably, when you started reading this article you were one of them. But a problem arises when that freedom collides with the freedom of others to use the Internet. That's exactly what congestion represents.

Imagine a world where some Internet service providers offer a deal, still for a flat price, but with a monthly congestion volume allowance. Note that this isn't a download allotment. In this scheme, you can download as much as ever. If you wanted to download ten videos continuously, as long as your peer-to-peer software used the new TCP with the weight set low, you wouldn't bust your allowance. Your video downloads would draw right back during the brief moments when intermittent flows came along with higher weights. But in the end, your video downloads would hardly finish any later.

On the other hand, your Web browser would set the weights high for all the little intermittent pages you browsed. It would be able to afford all these intense but brief flurries without using up much of your allowance.

Of course, we wouldn't all have to have the same allowance – server farms or heavy users could buy bigger congestion quotas, and light users could get Internet access with a tiny congestion allowance – for a lower flat fee than today's one-size-fits-all rates.

If congestion limits are so perfect, why don't Internet service providers use them? It turns out they can't limit congestion because it's hidden from them. As we've said, Internet congestion was intended to be detected and dealt with solely by the computers at the edge. So your computer sees all the congestion its traffic encounters across the Internet, but each provider doesn't—network equipment only sees its local congestion.

ISPs can easily see the bare volume a customer transfers even though they can't see *congestion* volume. So some limit the volume, in gigabytes, that each customer can transfer in a month, or during peak periods. As we've seen, that's neither necessary nor sufficient to solve the sharing problem. They don't need to limit volume in the moments when other computers aren't filling a link to capacity.

Imagine networks can see congestion, so they can limit it. And they can see congestion ahead in other networks, so they can reroute your data round it. And whenever a network can't reduce congestion ahead by rerouting, it knows that continued congestion represents real demand for more capacity—after all, the traffic causing it must come from people willing to draw down their congestion allowances.

Those networks in most demand can levy a penalty against the neighbouring networks who dump congestion into them—once they can see it. They can spend the cash raised on capacity upgrades to alleviate the congestion. True, their income from penalty charges will drop, but they can raise their capacity charges. If they don't upgrade, but a competing network does, they will lose the business altogether to the competing network. Back in the mid-1990s this competitive process was explained by Hal Varian, now Chief Economist at Google. It turns congestion into a commodity with the same price as the cost of the capacity needed to alleviate it.

David Clark, a founding father of the Internet, once famously said, "We figured out how to route bits, but we never figured out how to route money." Now we know why—we never figured out which bits routed into congestion. Once your computer has to use up your allowance to send bits into congestion, those bits can tell network operators exactly where customers want more capacity and carry just enough cash to buy it.

Even better, you won't have to suffer the impairment of actual congestion, because we will be able to use the most recent change to the Internet protocol, which signals *approaching* congestion.

And you won't have to pay more—just as some of your flat fee today goes towards capacity upgrades, the cash flowing to the most congested links will be drawn from the same flat fee. But the existence of congestion limits will be enough to make software developers use your weighted TCP judiciously.

So what's the trick that reveals congestion so that limits can be enforced? And what changes would have to be made to the Internet? My team call our new scheme "re-feedback." Here's how it works.

Recall that today, congestion anywhere across the Internet is detected by the receiving computer which returns feedback to the sending computer. So they both see any congestion, but the networks between them can't rely on this feedback being honest, or even visible. Under re-feedback, the sending computer has to re-insert the congestion feedback it gets from the receiver back into the forward data flow, so on average packets say how much congestion can be expected up ahead.

But why would the sender reveal information that counts against its own limit? The deal with re-feedback is that any network equipment can discard packets if the information about expected congestion is persistently *less* than the actual congestion encountered on the path. So, if the sender under-declares, packets won't get through.

We managed to arrange re-feedback so existing network equipment wouldn't have to be changed, except of course the boundary equipment enforcing the limits. But in order to reveal congestion to this equipment, we have to tweak the Internet protocol itself —the "IP" part of "TCP/IP. There was no way round

this, given the problem is embedded deep within the basic design assumptions of the Internet. There is one last unused bit in the header that IP attaches to every packet of data, and that one bit is all we need. By 2005, we had documented all the technical details in a proposal to the IETF—the Internet standards body.

At this point, the story gets personal. From previous experience of getting IP changed, I now set aside the next seven years in my diary. However, I knew changing IP was only half the battle. I was proposing a two-part change: i) re-feedback in IP and ii) weighted TCP on the end systems. I had set myself the task of challenging TCP-friendliness—equality of flow rates—which, for my audience, was the very definition of fair sharing on the Internet.

The change to IP had to be first; otherwise everyone would just set their weighted TCPs to the maximum. So I decided to conveniently omit any mention of weighted TCP. Instead I bigged up some other motivations for adding re-feedback to IP. I even showed how re-feedback could enforce *equal* flow rates—pandering to my audience's faith while denying my own.

But by skirting round the TCP-friendliness problem, I just boxed myself into a corner. I looked like yet another mad researcher pushing a solution without a problem.

After a year of banging my head against a wall at the IETF, my anger exploded. I wrote a polemic dismantling the religious dogma that asserted equal flow rates were “fair.” Fortunately my colleagues got me to tone it down before posting to the IETF, sufficiently at least to be invited to present it at a plenary session in San Diego late in 2006. The next day, a non-binding straw poll of the large audience showed widespread doubt about using TCP-friendliness as a definition of fairness. Elwyn Davies of the Internet Architecture Board emailed me saying, “you have identified a real piece of myopia in the IETF.” Finally the delusions that govern how much Internet capacity you get were starting to erode.

I was hardly the first to dispel these myths. In 1997 Prof Frank Kelly of the University of Cambridge used awe-inspiringly elegant and concise maths to prove the same weighted sharing would maximise the total value derived from the whole Internet.

To create the right incentives to flip the weights, he used a simple and elegant scheme; but therein lay the problem. Without re-feedback, congestion could only been seen where the *receiving* computer attached to the Internet. So Kelly's incentive mechanism relied on charging consumers a dynamically varying price for receiving packets that had caused congestion, rather than rationing congestion at source. Everyone balked at congestion charging, believing it would be too unpredictable for Internet consumers to swallow.

In the ensuing years, Kelly went on to serve as Chief Scientific Advisor to the UK Department for Transport just after the London congestion charge was introduced—an equally emotive subject. At the same time as Kelly was limiting congestion on Britain's roads, my team was designing re-feedback to limit congestion on the Internet, building around his earlier ideas, but with flat fees *not* dynamic pricing.

Back in the late '90s, the pricing scheme Kelly had proposed seemed to blind the Internet community to all the other insights in his work—particularly the message that equalising flow rates was not a desirable goal. TCP’s magic sauce had somehow made the Internet immune to economics. Everyone, rich and poor, *seemed* to be treated equally. Allowing economics in to upset this innocent world was too risky to even contemplate—it might also upset everyone’s voluntary use of TCP, the only protection against the ever-present danger of global congestion collapse.

Only a small minority of researchers had allowed themselves to realise that the Internet sharing problem had little to do with equalising data rates. Just as limiting global warming has little to do with the speed your car goes, unless you also take account of how long you drive at that speed. Even if we wanted equality, TCP wouldn’t achieve it by equalizing packet data rates, because data rates are the *wrong metric* to equalize. The correct metric is congestion volume—the CO_2 of the Internet.

My unenviable role now is to be the dripping tap – the one who repeatedly demands vigilance against the ways of the old religion. But the next immediate task is to build support for a standards working group at the IETF to reveal the missing congestion information in the Internet. The chosen mechanism may be re-feedback, but I won’t be fussed if something better emerges.

ABOUT THE AUTHOR
Bob Briscoe is Chief Researcher at BT.

FURTHER INFORMATION
Further papers on Internet fairness and on the re-feedback solution can be found at
<http://www.cs.ucl.ac.uk/staff/bbriscoe/projects/refb/>

