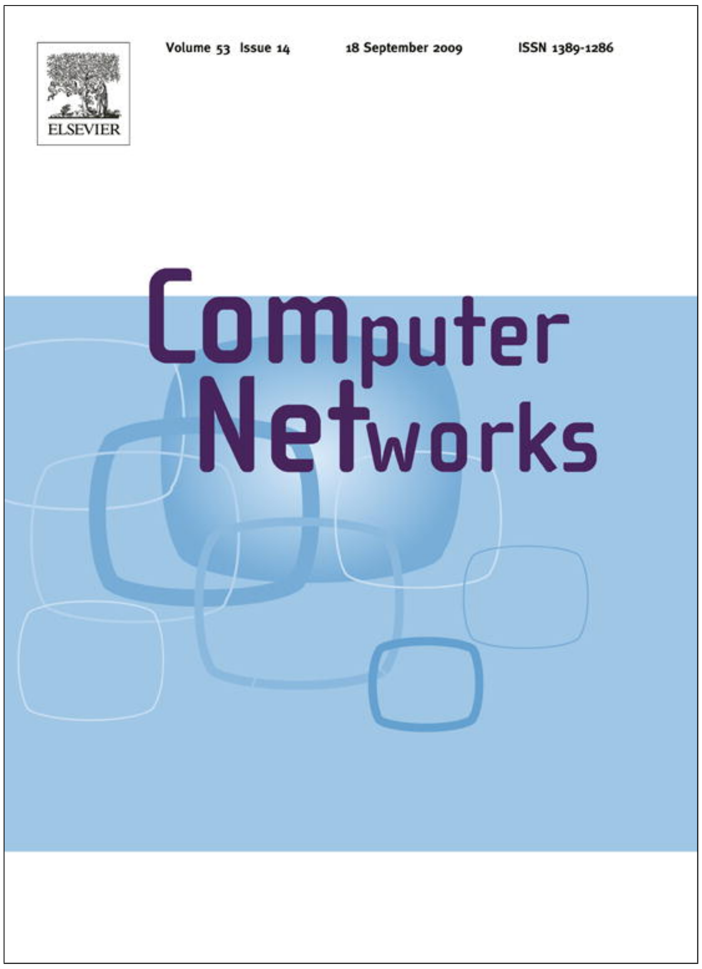
Volume 53 Issue 14
18 September 2009
ISSN 1389-1286
ELSEVIER
Computer Networks

Computer Networks 53 (2009) 2444–2457

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Transport-independent fairness

B. Behsaz, P. Gburzynski, M. MacGregor *

Dept. of Computing Science, University of Alberta, Edmonton, Alberta, Canada

A R T I C L E   I N F O

Article history:
Received 26 August 2008
Received in revised form 13 January 2009
Accepted 28 April 2009
Available online 12 May 2009

Responsible Editor: G. Morabito

Keywords:
TCP
P2P
Congestion control
Optimization
Convex programming

A B S T R A C T

The Internet relies on cooperative endpoints to react to signals from the network that congestion is occurring. In particular, TCP interprets packet loss as a signal of congestion. However there are many new non-cooperative protocols in use which attempt to exploit the network aggressively and do not reduce their demands when the network signals congestion. We propose the aggregate control of "fluxes" defined by policies at individual routers. Each router can then calculate an optimal allocation of bandwidth to each flux contending for a given output link. We propose a combined hill climbing and convex programming method for this optimization, which we call HCCP. HCCP is designed to punish greedy fluxes rather than just regulating them: such fluxes may find their bandwidth allocation reduced to zero if they are sufficiently aggressive. Our results show that HCCP is effective at regulating a wide range of rather generally characterized transport protocols. We explore the use of both throughput maximization and proportionally fair allocation and recommend the latter because the former often leads to the situation where one or more fluxes receive zero bandwidth.

1. Introduction

The network layer of the Internet as implemented by IP is connectionless. This means that all packets appear to it as independent datagrams. The concept of a session, understood as a sequence of related packets, is implemented at the edges of the network and formally the network layer can be completely oblivious of that concept. This simple paradigm has both advantages and drawbacks. Its obvious advantage is simplicity: routers can be made memory-less with respect to the packets they are forwarding. Their sole responsibility is to map destination addresses of incoming packets to outgoing links. The primary drawback is the lack of accountability of datagrams for their sessions. For example, during a congestion event a router may find it difficult to discard packets in a way that would reflect the extent to which particular sessions are responsible for the overload. In other words, the router may find it difficult to be fair.

In the early days of networking, when the networks were separate and confined to small communities with shared interests and goals the simple paradigm of IP was quite adequate. These days, however, the network is global and its users compete for limited bandwidth. One of the major problems facing Internet Service Providers (ISPs) is that a large proportion of the traffic in their networks is transported by aggressive, greedy protocols which serve the needs of disparate communities. Consequently there is a growing need for fairness mechanisms within the network core – mechanisms that can identify users exceeding their fair share of bandwidth and reduce the number of their packets in the network without hurting more considerate citizens. Most of the schemes attempting to address this issue focus on identifying transport layer sessions with the intention of treating them as the "users" whose bandwidth shares and behavior are subsequently assessed and policed [13,28,29,34]. In particular, TCP sessions are easy to identify and monitor because of their explicit connection-oriented nature.

The idea of viewing TCP sessions as the basic units which contribute to network load dates from the infamous

* Corresponding author. Tel.: +1 780 492 7434; fax: +1 780 492 1071.
E-mail address: mike.macgregor@ualberta.ca (M. MacGregor).

1389-1286/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2009.04.011

collapse of the Internet in 1988–1989 and its salvage by Jacobson [19]. The primary objective of any social (i.e., compliant to [19]) implementation of TCP is to navigate towards a state whereby random packet losses at a congested router will result in approximately the same fraction of bandwidth being received by each of the competing greedy sessions. This would have solved the fairness problem in an ideal world where:

(1) All TCP implementations are compliant and preferably identical.
(2) Users are synonymous with TCP sessions so that the fair treatment of individual TCP sessions translates into the fair treatment of actual users.
(3) There is no traffic in the network other than TCP sessions.

All three conditions would have to hold simultaneously; however, none of them in fact holds in today's Internet. First, as the behavior of a TCP session is solely up to the edge host there is little that a router can do to effectively enforce user compliance. The only possible approach is to apply some heuristics to the observed packet arrival process in order to detect misbehaving flows and then penalize those flows by dropping their packets [20]. But this is a cat and mouse game: a session familiar with those heuristics may be able to dynamically adjust the rate of its packets as to fool the router into giving it disproportionately more bandwidth.

Even more importantly, TCP sessions are not synonymous with users. In particular, an end-to-end application needing a lot of bandwidth can easily open multiple TCP sessions. And lastly, all the measures aimed at policing TCP sessions can be circumvented by resorting to UDP and implementing sessions within the application [14,16].

Network operators and users have adopted the idea of Service Level Agreements (SLAs) in which separate domains are individually responsible for enforcing the agreed allocation of resources to support the SLA. In this case, the requirements are known very specifically at the edge of the network, and rather than individual "hosts" the clients of the proposed scheme are other ISPs. Knowledge of the service requirements at an edge router alone, however, is not sufficient to ensure that the rest of the domain will respect the intent of an individual SLA, especially when presented with flow aggregates from other customers in other domains terminating on other edge routers whose resource requirements are supposedly guaranteed by some other SLA. Our proposal represents an effective mechanism that can be used to police the competing interests of multiple customers in a complex network.

For a router, the problem of identifying the actual *users* of the network is difficult and not even particularly well defined without introducing some additional concepts. The primary problem is that (as explained above) the perceptible attributes of packets, as seen by the router, do not allow it to authoritatively decide which "user" those packets belong to. Here by "user" we understand an entity that would be meaningful from the viewpoint of a globally acceptable fairness measure. The behavior of such an entity should also be controllable: a de-facto single user should not be able to fake multiple identities to receive extra bandwidth.

A truly workable mechanism for global fairness must be implemented within the network core, as opposed to at the edge. Such a mechanism must be completely transport-independent, in the sense that it cannot be based on formal transport-layer sessions, be they explicit TCP flows or some conceptual streams derived from packet parameters (e.g., source/destination addresses, ports, or patterns spotted in the IP payload). This is because those parameters can always be modified by the source in order to evade control. In particular, a collusion of hosts within a collaborating group of users may create a configuration whereby different parts of the same sessions follow different paths in the network, as in BitTorrent [25,33]. A reliable fairness enforcement scheme should be comfortable with such complications, which should not be perceived as cases of network abuse, but rather embraced as natural and creative ways of harnessing the power of the connectionless core of the network. If properly designed, such a scheme will also effectively obviate any need for explicit *calls* into the otherwise beautiful paradigm of connectionless operation and go a long way towards providing quality of service, thwarting DoS attacks, and generally solving the problems caused by disparity and unaccountability of bandwidth allocation.

In this paper, we propose a global scheme for fair allocation of network resources using mechanisms at the routers, rather than relying on the transport behavior of endpoint applications. We address this problem at a level of aggregated flows to avoid the usual fallacies associated with attempts to police individual transport layer sessions. The definition of our flows is dynamic and hierarchical so as to account for the different policies assumed by different hosts and routers. Its role is to meaningfully combine those policies and create a natural way of interpreting them as a definition of a globally fair network state.

One central issue is the identification of unfair flows, i.e., those demanding more bandwidth than their fair share, and differentiating their treatment. A second related issue is to make sure that those flows that consistently demand more than their fair share and behave in an uncooperative manner are not able to cheat the policing mechanism. We call such flows *greedy* to differentiate them from those that are only temporarily unfair. The idea is to discriminate more heavily against greedy flows than against those which are only temporarily unfair but which respond cooperatively to network regulation. The way the unfair flows are penalized encourages them to converge to their fair share. This does not assume any particular implementation of those flows at the hosts such as a compliant TCP implementation. The router does not try to guess at or emulate the mechanism used by the hosts in response to packet loss. It merely drops the excess packets of the unfair flow in a certain systematic manner so as to convey a certain universally meaningful *message* to the application. In a nutshell, the message says something like this: "*The more you try to exceed your fair share, the less of the actual bandwidth you are going to get.*" The allocated share of bandwidth is not proportional to the flow's apparent demand, but instead it is a decreasing function of the demand in excess of the fair share. Thus, it is simply in the interest of the flow to be cooperative, as otherwise its effective bandwidth will be reduced to less than its actual fair share.

2. Flow aggregates: fluxes

As defined for the purposes of our scheme, and as viewed by a router, a flow is a meaningful recipient of bandwidth from the viewpoint of its fair allocation. To distinguish such *flows* from traditional flows such as TCP sessions or source–destination pairs we shall call them *fluxes*. This concept is similar to the idea of *classes* as proposed in [9]. However that work dealt only with network gateways, while the present proposal extends into the interior of the network and includes a specific algorithm for optimal allocation of resources.

Taking this point of view enables us to aggregate multiple flows within a flux, which is prerequisite for a meaningful definition of fairness in the context of poor representability of *users* by flows. The use of fluxes also enables scalability of this proposal. Other well-known techniques such as class-based queueing (CBQ) use a similar approach.

As we pointed out in Section 1, a router is essentially unable to identify fluxes without external hints. Needless to say, to carry any objective value, such hints cannot be based on user declarations, but must account for the hierarchical structure of network interconnections and authorities. For example a system administrator may have the authority to rank fluxes within a given host but the edge router to which it is attached may view the complete collection of incoming fluxes from that host as a single flux falling under the authority of the router's administrator. This way the router can be concerned with the fair allocation of its bandwidth to the hosts, while the hosts deal with fairness with respect to their users.

Let us define a flow as the smallest component of a flux that can be discerned by a router. Such a flow corresponds to a single transport layer "session" and is described by the 4-tuple $\langle S, P_s, D, P_d\rangle$ where S is the source IP address, P_s is the source port, D is the destination IP address, and P_d is the destination port. Let A be the set of all flows passing through a router. By a flux we understand a pair $F = [\Phi, q]$ where $\Phi \subseteq A$ and $q > 0$ is a number. In simple words, a flux is a container for a subset of the set of flows passing through the router. Each flux has an associated rank q. The role of q is to provide a means for implementing intentional bias reflecting the subjective importance of each flux.

A router builds a description of the fluxes passing through it. This description is derived from flux descriptors communicated to the router by its neighbors and its own flux policy. Edge routers receive flux descriptors from hosts. An uninformed "vanilla" router, which has acquired no descriptors from the outside and defines no internal flux policy defines a single flux denoted $[\langle *, *, *, *\rangle, 1]$. The asterisks serve as wildcards and provide a shorthand notation for all possible values of the respective flow parameters. Thus, $\langle *, *, *, *\rangle$ stands for the set of all flows arriving at the router and the last element is the blanket rank assigned to those flows. The default flux description of an "uninitialized" router simply says that all flows that the router ever sees belong to a single flux; thus, there is no regulation of competition for bandwidth among those flows.

The set of fluxes perceived by any router can be formally described using a language derived from the above wildcard notation. Specifically, fluxes can be equivalenced with sets of 5-tuples where any of the first four items can be wildcards. An incoming flow is qualified to the flux whose description includes the 5-tuple that most closely matches the flow's parameters. Let us ignore any possible ambiguities assuming that fluxes are always specified in an unambiguous way. This assumption has the status of a "sanity postulate" for all practical cases.

The allocation of bandwidth at a router deals with the router's output links and is carried out on a per-output-link basis. The router identifies the different fluxes competing for a given output link and views them as the contenders that should be treated fairly.

Consider the situation depicted in Fig. 1. Each of the three hosts connected to the edge router R1 defines its description of fluxes and passes it to R1. The descriptions of fluxes prepared by, say H1, reflect the host's idea of how the flows of its users should be partitioned into streams that will then compete for bandwidth in a fair manner. Note that H1 is not aware what other hosts may be connected to the edge router: it is R1's responsibility to provide for a fair treatment of fluxes arriving from all its directly-connected hosts according to the router's policy.

2.1. Ranking fluxes

To provide for a fair treatment of the traffic arriving from its hosts, R1 (in Fig. 1) will normalize the different fluxes to their population. For illustration, consider the trivial network shown in Fig. 2 and suppose that host H1

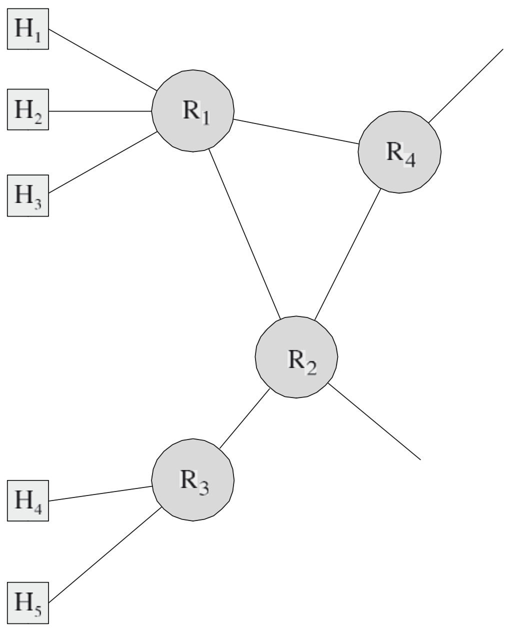

Fig. 1. A network fragment.

defines three fluxes, host H2 two fluxes, and host H3 four fluxes. If R1's policy is to treat the three hosts equally, then each flux of H1 will receive the rank of $1/(3 \cdot 3) = 1/9$, each flux of host H2 will receive the rank of $1/(3 \cdot 2) = 1/6$, and each flux of H3 will be ranked $1/(3 \cdot 4) = 1/12$. These numbers directly represent the fraction of the router's global bandwidth that will be assigned to every flux. Let us note that they all add up to one.

The situation is slightly complicated by the fact that the hosts (and more generally neighbors, which can be other routers) of R1 may pre-rank the fluxes themselves. Let us consider a general case and denote the collection of all fluxes received by a router from its m neighbors as $F_i^j, 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n_i$, where n_i is the number of fluxes defined by neighbor i. Let q_i^j denote the rank of flux F_i^j as assigned by neighbor i. We shall assume that $\sum_{j=1}^{n_i} q_i^j = 1$, otherwise the router can renormalize the ranks to fulfill this condition.

The router's policy is described by the ranks of its neighbors, denoted $r_i, 1 \leqslant i \leqslant m$, such that $\sum_{i=1}^{m} r_i = 1$. The ranks of the incoming fluxes are rescaled with respect to the router's preference policy as $\rho(F_i^j) = r_i \cdot q_i^j, 1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n_i$, where $\rho(F_i^j)$ is called the effective policed rank of flux F_i^j at router R.

The rather simple idea inherent in the above procedure is to arrive at a ranking of fluxes that would allow a router to respect the ranks passed to it by other authorities, while superimposing its own ranking policy. If a given incoming flux were entirely passed over a single output link it would be easy to propagate the rank information. Specifically, if $\{G_u^k\}, 1 \leqslant k \leqslant n_u$ is the set of all fluxes sent on link u, the router would pass to the neighbor connected to it via link u the ranks:

$$s_u^k = \frac{\rho(G_u^k)}{\sum_{j=1}^{n_u} \rho(G_u^j)}, \tag{1}$$

reflecting the router's effective policed ranks of the respective fluxes. This would be the case, for example, if a single flux were not allowed to include traffic addressed to different destinations as that would result in partitioning the flux over multiple output links at the router. While one might consider it useful to restrict the notion of flux this way, such a restriction would be useless as the basis of a meaningful and enforceable concept of fairness. This is because various proxies posing for destinations and forwarding the received traffic to other hosts could circumvent any restrictions imposed on source–destination streams. Notably, most QoS schemes operating today ignore this issue and are thus susceptible to bandwidth extortion by greedy colluding users as is common in P2P systems.

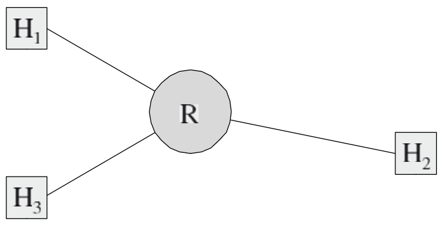

Fig. 2. A trivial network.

One obvious assumption of the proposed scheme, which has not been explicitly mentioned so far, is that the description of fairness rules in terms of the fluxes and their ranks maintained by hosts and routers is generally dynamic and thus requires a periodic exchange of information among the neighbors. This is reminiscent of the protocols for propagating routing or QoS information (OSPF [30], BGP [35], RSVP [43]). In the face of the inherent dynamics of our approach, routers can additionally monitor the way their fluxes are split over multiple outgoing links and update their ranks accordingly. Then at certain intervals, and/or based on threshold triggers, updates with respect to the last declared configurations can be propagated to the neighbors, where they may affect the neighbors' configurations of flux ranks, and so on.

The specific implementation of the mechanism for exchanging information about fluxes and their rankings must, of course, allow for incremental adoption of this proposal. That is, "neighbor" must be defined in a sense similar to that used in BGP, where routing information is exchanged between BGP-speakers which are not necessarily adjacent. Intervening nodes are not required to participate in the protocol.

Similarly, we have posed this proposal within the common context of the point-to-point links commonly present in the high-bandwidth "core" sections of the Internet. We will continue with an explanation based on that context because we believe it makes our proposal clearest. However, it is certainly possible to define fluxes with respect to other granularities of aggregation. For example, outgoing fluxes could be ranked separately depending on their membership in distinct VLANs carried by the same physical interface. In the case of a broadcast medium, fluxes could be defined based on individual destination addresses or subnets.

Note that from the viewpoint of a fair assessment of the different parts of a given flux that are directed to different output links, a router can look no further than the incoming traffic. This is important because the different segments of a flux can be subject to different drop rates when they are queued for forwarding at different output links, and thus their output rates need not match the input rates, even in relative terms. Essentially, each portion of an input flux directed to a given output link becomes a new flux. The rank contribution of that portion is weighted by its percentage of the original input flux.

Let $\gamma_u(F_i^j, G_u^k)$ denote the normalized fraction of flux F_i^j directed to output link u and indexed as the k'th flux going out over that link. The word "normalized" means that $\sum_{u=1}^{m} \gamma_u(F_i^j, G_u^k) = 1$ over the m output links at the router. Then:

$$\Psi_u = \{F_i^j | \gamma_u(F_i^j, G_u^k) \neq 0\}$$

is the set of fluxes that feed into output link u. The rank of (output) flux G_u^k communicated by the router to its neighbor over link u is calculated as:

$$\rho(G_u^k) = \frac{\rho(F_i^j) \cdot \gamma_u(F_i^j, G_u^k)}{\sum_{F_i^j \in \Psi_u} \rho(F_i^j) \cdot \gamma_u(F_i^j, G_u^k)}. \tag{2}$$

The router determines the value $\gamma_u(F_i^j, G_u^k)$ by monitoring the input flux and measuring its fraction routed over link u. Thus, $\gamma_u(F_i^j, G_u^k)$ is dynamic and will tend to change over time.

2.2. Identifying unfair fluxes

At this point, we can assume that a router knows its population of fluxes, specifically its output fluxes and their associated ranks. We shall focus on the operation of a single output link u. Our goal is to implement a fair sharing of that link among the fluxes in Ψ_u.

Recall that n_u is the number of fluxes in Ψ_u. As part of determining the partitioning of the input fluxes over the output links the router calculates the perceived dynamic input rate of every flux in Ψ_u. Let x_u^k denote the offered rate of each flux in Ψ_u. That is, $x_u^k = \gamma_u(F_i^j, G_u^k) \cdot \lambda_i^j$ where λ_i^j is the arrival rate of flux F_i^j from neighbor i. Let B_u stand for the capacity of link u. If $\sum_{k=1}^{n_u} x_u^k \leqslant B_u$ there is no problem: the link's capacity leaves enough room to accommodate all fluxes with no loss. The fairness enforcement scheme is only required if $\sum_{k=1}^{n_u} x_u^k > B_u$. Let us suppose that this is the case.

The simple goal of the fairness enforcement scheme is to produce a prescription for dropping packets belonging to different fluxes. The first step consists in isolating the unfair fluxes from the ones that use their fair share. To this end, the router executes the following algorithm:

```
$C = \{1, \ldots, n_u\};$
$s = n_u;$
$B_u^{\text{cont}} = B_u;$
while $(\exists k \in C | x_u^k < B_u \cdot \rho(G_u^k))\{$
    $C = C - \{k\};$
    $B_u^{\text{cont}} = B_u^{\text{cont}} - x_u^k;$
    $s = s - 1;$
}
```

Note that from Eq. (2) we have that $\sum_{k=1}^{n_u} \rho(G_u^k) = 1$. The usage of $\rho(G_u^k)$ in the algorithm above explains the interpretation of the rank of a flux: it directly determines the proportion of the link's bandwidth that the flux should receive in a fair allocation under contention. The algorithm calculates two values: B_u^{cont}, which is the bandwidth subject to contention after all fluxes that remain within their fair shares have been allocated their requested bandwidth, and C, which is the set of indexes identifying the unfair fluxes.

Let $C = \{c_1, \ldots, c_s\}$. One can easily see that $\sum_{k=1}^{n_u} x_u^k > B_u$ implies that C is nonempty. Indeed, suppose that $\sum_{k=1}^{n_u} x_u^k > B_u$ and that $x_u^k \leqslant B_u \cdot \sum_{k=1}^{n_u} \rho(G_u^k) \forall k \in \{1, \ldots, n_u\}$, which is the necessary condition for C to be empty. Then we have $\sum_{k=1}^{n_u} x_u^k \leqslant B_u \cdot \sum_{k=1}^{n_u} \rho(G_u^k)$, which is a contradiction because $\sum_{k=1}^{n_u} \rho(G_u^k) = 1$.

2.3. Fair allocation of bandwidth

We would like to allocate bandwidth fairly while not leaving the network open to abuse by endpoint applications which try to obtain more than their fair share. The ability to do so no matter what strategy is adopted by the endpoint applications is our goal. We call this the "transport-independent fair bandwidth allocation problem." We have previously defined the fair bandwidth allocation for each outbound portion of an individual flux, namely $b_u^k = B_u \cdot \rho(G_u^k)$. In this section we define our idea of a *fair maximum bandwidth function*. This function enables us to ensure fair treatment amongst fluxes.

Definition 2.1. A function $m(x_i, b_i)$ of incoming rate x_i and fair share b_i is a fair maximum bandwidth function if:

(1) For any two fluxes i and j such that $x_i/b_i = x_j/b_j$ we have $m(x_i, b_i)/b_i = m(x_j, b_j)/b_j$.
(2) For any two fluxes i and j such that $x_i/b_i < x_j/b_j$ we have $b_i/m(x_i, b_i) \geqslant b_j/m(x_j, b_j)$.

The first condition ensures that if two fluxes have the same ratio of offered load to fair share then they will be given the same ratio of maximum bandwidth to fair share. The second condition ensures that a higher ratio of offered load to fair share does not increase the ratio of maximum bandwidth to fair share. This implies that a fair maximum bandwidth function is an upper bound on the bandwidth that a router will allocate to a flux.

Definition 2.2. A bandwidth allocation mechanism which allocates bandwidth $\hat{x}_I$ to a flux F_i is a transport-independent fair allocation mechanism with respect to a fair maximum bandwidth function $m(x_i, b_i)$ and ranks r_i if it satisfies the following conditions for all links u:

(1) $\sum_{k=1}^{n_u} x_u^k \leqslant B_u$.
(2) $x_u^k \leqslant b_u^k \Rightarrow \hat{x}_u^k = x_u^k$.
(3) $x_u^k > b_u^k \Rightarrow \hat{x}_u^k \leqslant m(x_u^k, b_u^k)$.
(4) If an unfair flux changes its offered load from $x_u^k(t_1) > m(x_u^k, b_u^k)$ to $x_u^k(t_2) > x_u^k(t_1)$, and there are no changes in the offered loads of any other fluxes, its allocated rate changes from $\hat{x}_u^k(t_1)$ to $\hat{x}_u^k(t_2) \leqslant \hat{x}_u^k(t_1)$.

The first condition constrains the allocated rates to the capacity of the link. The second condition says that a flux requesting its fair share will be allocated what it requests. The third condition says that a greedy flux (i.e., one exceeding its bandwidth limit) will be allocated no more than its maximum fair bandwidth. The fourth condition ensures that a flux cannot obtain more bandwidth (in excess of its fair share) by strategically increasing its offered load—that is, by being greedy. This guarantees transport-independence of allocated bandwidth.

There are many interesting fair maximum bandwidth functions that satisfy Definition 2.1. In fact, in some circumstances it may make sense for this function to allocate bandwidth greater than the defined fair share. This would give us the option to allocate more bandwidth to the unfair fluxes we prefer—those that yield higher utility. We may also define this function to yield allocations less than fair share. One possibility is:

$$m(x_u^k) = x_u^k (b_u^k / x_u^k)^\alpha, \tag{3}$$

where $\alpha \geqslant 1$ is a constant. This particular function ensures that fluxes cannot exceed their fair share.

To illustrate the behavior of the maximum fair bandwidth function in Eq. (3) we examine its behavior in the single-router environment of Fig. 2. In the following we use the four general types of fluxes listed in Table 1. These generic flux types were defined so that we could explore the behavior of the function independent of specific transport protocols. They are not meant to be accurate representations of specific protocols although their general characterizations as "rational", "greedy", etc. was chosen to convey their intent. In Table 1, x is the offered load, $\hat{x}$ is the bandwidth allocated, d is the actual demand the flux seeks to reach, and V_{RA} and V_{NG} are threshold values. For the following experiments we set $\alpha = 1.5, V_{RA} = 1.5$ and $V_{NG} = 3.0$.

First we test the behavior of a set of fluxes in the case where the capacity of all links is so low that all fluxes are allocated bandwidth less than their demand. We set $B_1 = B_2 = B_3 = 1000$ with the local policy described as $r_1 = 0.3, r_2 = 0.4$ and $r_3 = 0.3$. The set of fluxes is defined in Table 2.

The loads offered by a rational flux and a near-greedy flux are shown in Fig. 3. These are fluxes 1 and 3 on link 2. The rational flux has a fair share of 273 and converges relatively quickly to an offered load of this value by $t = 100$. The near-greedy flux has a fair share of 182 but persists in trying to achieve a much higher rate and displays an offered load which oscillates between about 190 and 3250. Fig. 4 shows the results of applying the fair maximum bandwidth function to these two fluxes along with flux 6 (rational, fair share = 364). The rational fluxes both achieve their fair share and converge to offered load at that value resulting in zero drop rate. The near-greedy flux suffers an increasing penalty in the phase where it increases its offered load. It only achieves its fair share after decreasing the load it offers. The other near-greedy flux traversing this link, flux 9, is treated similarly: its carried load is bounded from above by its fair share. Clearly, despite the relatively simple form of the maximum fair bandwidth function, it is effective in enabling cooperative fluxes to achieve their goals while preventing greedy fluxes from dominating the network.

A second set of results was obtained by changing the form of the fair maximum bandwidth function to:

$$m(x_u^k) = x_u^k - \beta \cdot (x_u^k - b_u^k) \quad \forall x_u^k > b_u^k. \tag{4}$$

For the following results we chose $\beta = 1.05$. We used the same set of fluxes as in the first scenario but different fractions of the incoming fluxes were distributed to the outgoing links (see Table 3). The capacity of link 3 was increased to B_3 = 10,000 so that we could examine the behavior of a mix of fluxes on a non-bottleneck link. In Fig. 5 we present the carried traffic for link 2 which is still a bottleneck link. On link 2 the rational flux converges fairly quickly to its fair share of 600 and also reduces its offered load to that value, thus achieving a zero drop rate. In contrast the near-greedy flux shows an offered load which oscillates between about 1100 and 2200. It achieves a maximum carried load of just 365 while its fair share is actually slightly higher, at 400. Link 2 is operating at over 90% utilization with no impact from the near-greedy flux on the rational flux.

The behaviors on the non-bottleneck link are quite interesting (see Fig. 6). Again, the rational flux converges fairly quickly to its fair share. The near-rational flux takes longer to converge as a result of starting out at an offered load of 1500 which is slightly higher than its fair share of 1410. After cutting its offered load in half, its rate gradually increases towards its fair share. The two near-greedy fluxes nearly achieve their fair shares, oscillating at carried loads just slightly below them. However, they suffer significant and oscillating drop rates. The greedy flux behaves in a completely unreasonable fashion despite the fact that its fair share of 3530 is a large proportion of its total demand of 4000. As the fair maximum bandwidth function tries to persuade this flux to reduce its offered load, the flux consistently increases what it offers in an attempt to somehow exploit the network. The net result is that by time 150 the greedy flux is achieving a carried load of zero because of the extreme level to which it has increased its offered load. Link 3 is quite successfully carrying a mix of fluxes of

Table 1
Generic flux types.

Type	Description
Rational (RA)	$b \geqslant x \Rightarrow x = x + 1$ $(b < x) \wedge (x/b \leqslant V_{RA}) \Rightarrow x = x - 1$ $(b < x) \wedge (x/b > V_{RA}) \Rightarrow x = x/2$
Near-rational (NR)	$b \geqslant x \Rightarrow x = x + 1$ $b < x \Rightarrow x = x/2$
Greedy (GR)	$b \geqslant x \Rightarrow x = d$ $b < x \Rightarrow x = x + d - b$
Near-greedy (NG)	$b \geqslant x \Rightarrow x = d$ $(b < x) \wedge (x/b \leqslant V_{NG}) \Rightarrow x = x + d - b$ $(b < x) \wedge (x/b > V_{NG}) \Rightarrow x = x/2$

Table 2
Fluxes used in first scenario.

Flux	Source	Type	Demand	Incoming rank	Link 1 fraction	Link 2 fraction	Link 3 fraction
1	1	RA	3000	0.3	–	1.0	0.0
2	1	GR	4000	0.5	–	0.0	1.0
3	1	NG	3000	0.2	–	1.0	0.0
4	2	NR	5000	0.5	1.0	–	0.0
5	2	NG	5000	0.5	0.0	–	1.0
6	3	RA	2500	0.4	0.0	1.0	–
7	3	NR	2500	0.3	1.0	0.0	–
8	3	GR	2500	0.1	1.0	0.0	–
9	3	NG	2500	0.2	0.0	1.0	–

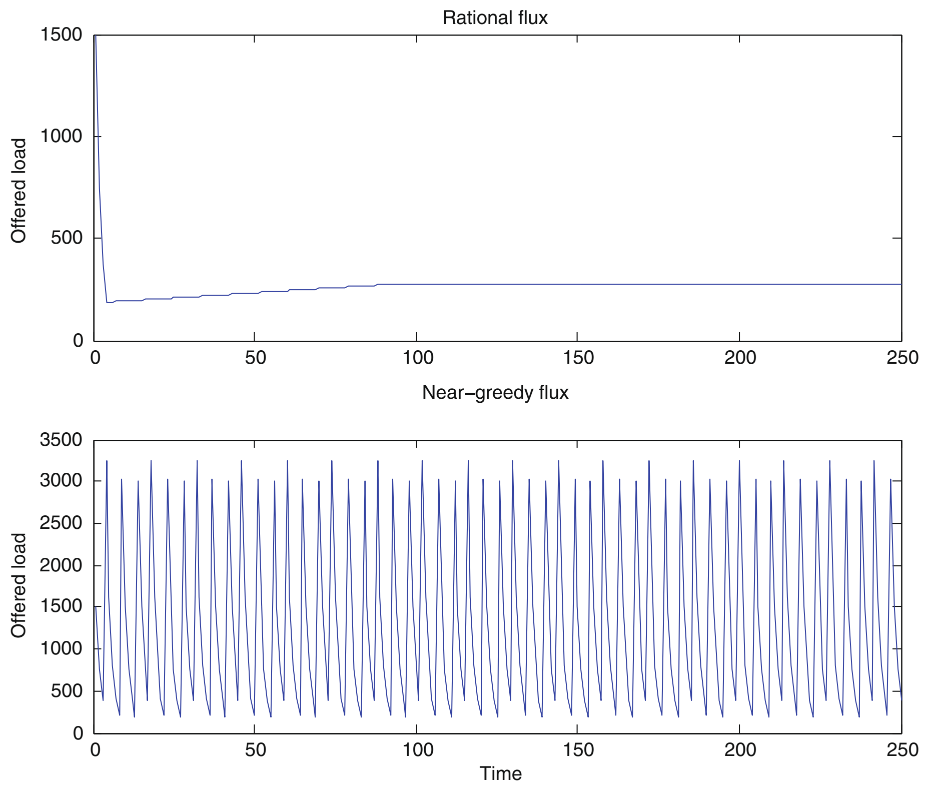

Fig. 3. Example offered loads.

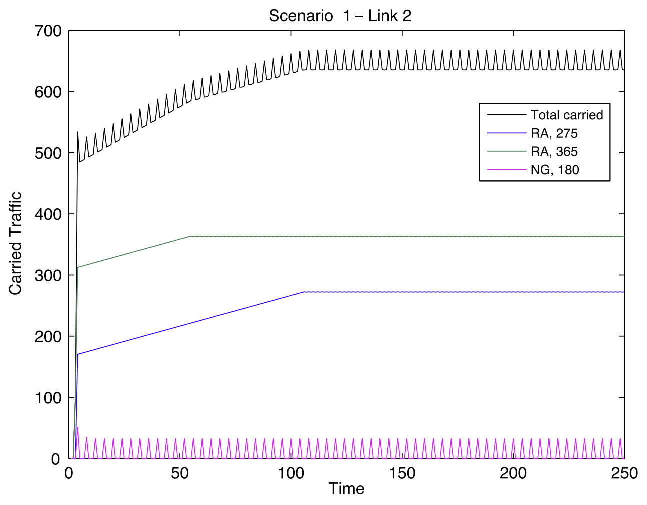

Fig. 4. Link 2 in the first scenario.

widely different behaviors. The rational and near-rational fluxes achieve their fair shares at zero or near zero drop rates. The near-greedy fluxes achieve their fair shares, but suffer significant drop rates. Notably, the greedy flux is completely thwarted in its efforts to escape control.

3. Optimal flux rankings

As demonstrated in the previous section, allocating bandwidth in the fashion proposed can be quite effective in regulating flows of widely varying characteristics. To use this mechanism the router must measure the offered load of each incoming flux, λ_i^j, and then calculate the offered output load at each link $x_u^k = \gamma_u(F_i^j, G_u^k) \cdot \lambda_i^j$. With the requisite values for x_u^k the router can find the subset of unfair fluxes for each output link. It will then forward the fair fluxes in their entirety and reduce the unfair fluxes to the

Table 3
Revised fluxes for second scenario.

Flux	Source	Type	Demand	Incoming rank	Link 1 fraction	Link 2 fraction	Link 3 fraction
1	1	RA	3000	0.3	–	0.5	0.5
2	1	GR	4000	0.5	–	0.0	1.0
3	1	NG	3000	0.2	–	0.5	0.5
4	2	NR	5000	0.5	0.7	–	0.3
5	2	NG	5000	0.5	0.3	–	0.7
6	3	RA	2500	0.4	1.0	0.0	–
7	3	NR	2500	0.3	1.0	0.0	–
8	3	GR	2500	0.1	1.0	0.0	–
9	3	NG	2500	0.2	1.0	0.0	–

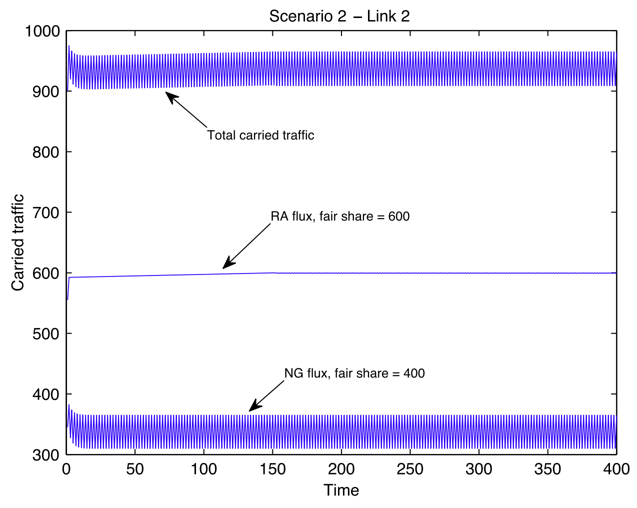

Fig. 5. Link 2 in the second scenario.

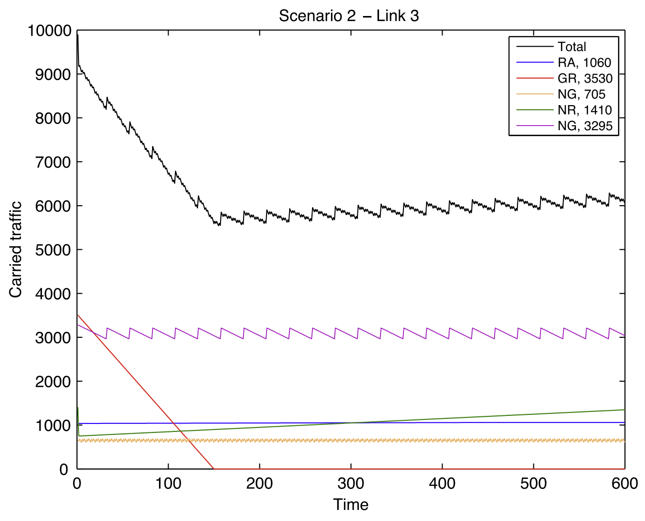

Fig. 6. A non-bottleneck link.

amount indicated by the fair maximum bandwidth function. The parameters which must be set are the value for α in the maximum fair bandwidth function and the ranks $\rho(G_u^k)$ for each flux. Suppose for the moment that α is constant and the same for all fluxes; then, the output ranks, $\rho(G_u^k)$, are the only free values. They are calculated from the incoming ranks $\rho(F_i^j)$ communicated by the router's neighbors and its own rankings r_i of those neighbors.

The r_i are the values through which the policies of the network operator are reflected, and through which end users will perceive the long-term average quality of service of the network. We propose to choose the r_i optimally so that the network achieves maximum utility while ensuring fair allocation of bandwidth and discouraging greediness. The scope of this optimization would, of course, be no greater than a single administrative domain. We are not proposing centralized optimization of the global Internet. The size of the optimization problem which will need to be solved will depend on the granularity of control desired: this will determine the number of fluxes at each router. The granularity of control will also determine the amount of state information that needs to be maintained at each router (the values of λ_i^j and $\gamma_u(F_i^j, G_u^k)$). The computational load of maintaining the router's state will depend on the interval over which the corresponding measurements are made and updated. Thus, there is a controllable trade-off between accuracy and load. With this introduction, we propose to solve the following problem.

3.1. Problem definition

Our goal now is to define an optimization tableau which will be used to find the best rankings for fluxes at each router in the domain. Of course there are many choices for an objective function. We will leave this somewhat free to begin with and simply define the objective function as optimizing network utility, where utility is a function of the bandwidth allocated to fluxes. Specific forms of the utility function will be selected subsequently. Let $U_i(\hat{x}_i)$ be the utility of allocating bandwidth $\hat{x}_i$ to flux F_i.

Assumption 3.1. The utility function $U_i(\hat{x}_i)$ is a continuously differentiable, increasing concave function.

We are interested in solving the following problem: Given a maximum fair bandwidth function $m(\cdot)$ find ranks r_i for the fluxes at a router and a transport-independent fair allocation mechanism that maximizes total utility at the router, $\sum U_i(\hat{x}_i)$.

3.2. The HCCP method

To solve our problem we use a hybrid method of hill climbing plus convex programming. We climb an objective function with respect to ranks to reach an ϵ-neighborhood of the global maximum.

Firstly, fix a vector $\mathbf{r} = (r_1, r_2, \ldots, r_m)$. With this we can calculate the fair bandwidth allocation b_i for each flux. Let $C_u = \{i : \gamma_u^i > 0 \wedge x_i > b_i\}$ be the set of unfair fluxes on link u and $N_u = \{i : \gamma_u^i > 0 \wedge x_i \leqslant b_i\}$ be the set of fair fluxes on that link. We employ the following convex program mechanism.

For each u and $i \in N_u$ we set $\hat{x}_i = x_i$. Then we calculate $m(x_i, b_i)$ for each $i \in \bigcup_m C_u$. We find the carried load for all the unfair fluxes by solving the following convex program:

$$\text{minimize} - \sum_{i=1}^{n} U_i(\hat{x}_i) \tag{5}$$

subject to:

$$\begin{array}{ll} \sum\limits_{i:\gamma_u^i>0} \hat{x}_i \leqslant B_u & 1 \leqslant u \leqslant m \\ \hat{x}_i \leqslant m(x_i, b_i) & i \in \bigcup C_u \\ \hat{x}_i \leqslant x_i & i \in \bigcup C_u \end{array}$$

The free variables in this program are the allocated bandwidths $\hat{x}_i$ for the unfair fluxes. While the objective function is convex all the constraints are linear. Thus standard algorithms can be applied. We prefer to use a *stable* algorithm, i.e., one that will return the same solution in every run in which that solution is optimal, even though some other optimal solutions may exist. We have the following lemma:

Lemma 3.1. *The convex program mechanism is a transport-independent fair allocation mechanism with respect to $m(\cdot)$ and r.*

Proof. It is clear that the convex program mechanism has the first three properties of Definition 2.2. It remains to show that the fourth property also holds.

Assume that an unfair flux F_i changes its offered load from $x_i(t_1) > m(x_i, b_i)$ to $x_i(t_2) > x_i(t_1)$ and that its allocated rate changes from $\hat{x}_i(t_1)$ to $\hat{x}_i(t_2)$. Due to the second property of the maximum fair bandwidth function in Definition 2.1 we have $m(x_i(t_1), b_i) \geqslant m(x_i(t_2), b_i)$. If $x_i(t_1) \geqslant m(x_i(t_2), b_i)$ then due to the second constraint in the tableau we have $m(x_i(t_2), b_i) \geqslant x_i(t_2)$ which yields $\hat{x}_i(t_1) \geqslant \hat{x}_i(t_2)$. If $x_i(t_1) < m(x_i(t_2), b_i)$ then the optimal solution before the change is still feasible in the new program. Clearly this solution is optimal and the *stable* program we use will return the same solution. Consequently in that case we get $\hat{x}_i(t_1) = \hat{x}_i(t_2)$. Thus we have $\hat{x}_i(t_1) \geqslant \hat{x}_i(t_2)$ as desired. □

This lemma has the following trivial corollary:

Corollary 3.1. *Among transport-independent fair allocation mechanisms with respect to $m(\cdot)$ and r the convex program mechanism maximizes $\sum_{i=1}^{n} U_i(\hat{x}_i)$.*

Proof. The allocation found by any transport-independent fair allocation mechanism is a feasible solution for our convex tableau. Thus the convex program mechanism yields the maximum value of $\sum_{i=1}^{n} U_i(\hat{x}_i)$ amongst all transport-independent fair allocations. □

This corollary confirms that if we know the optimal ranks then we can solve our convex tableau accurately. However, finding the optimal ranks to begin with is computationally difficult. We choose to conduct a local search with hill climbing. Given a fair maximum bandwidth function $m(\cdot)$ our hybrid algorithm HCCP is as follows:

```
(1) Choose an initial $\mathbf{r} = (r_1, r_2, \ldots, r_m)$ such that $\sum_i r_i = 1$. Let the value of the objective function for
    our convex tableau with this choice for $\mathbf{r}$ be $v^{(0)}$.
(2) **While** progress $> \epsilon$ **Do**
    • Generate several valid neighbor vectors of $\mathbf{r}$, say $r^{(1)}, \ldots, r^{(k)}$.
    • Use the convex program mechanism to find the respective values of the objective function $v^{(1)}, \ldots, v^{(k)}$.
    • Let $a = \mathrm{argmax}_i v^{(i)}$. If $v^{(0)} < v^{(a)}$ then update $\mathbf{r}$ with $r^{(a)}$.
    • Progress $= v^{(a)} - v^{(0)}$.
(3) Return $\mathbf{r}$ and the allocations $\hat{x}_i$ given by the convex program mechanism.
```

We have the following theorem for this hybrid algorithm:

Theorem 3.1. *The HCCP algorithm asymptotically converges to a solution that solves the transport-independent allocation problem with maximum error ϵ. That is, the value of the objective function for this allocation is at most ϵ less than the global optimum.*

Proof. By Assumption 3.1 $U_i(\hat{x}_i)$ is a concave function. Therefore $\sum_{i=1}^{n} U_i(\hat{x}_i)$ is also a concave function. Thus it has one local maximum and the local maxima of the objective function are also global maxima. We know that the hill climbing method asymptotically converges to a solution at most ϵ less than one of the local maxima, which in this case is a global optimum. Thus, Corollary 3.1 completes the proof. □

4. Experimental studies of HCCP

In this section we present experimental results for HCCP. The theoretical results presented in the previous section guarantee reasonable performance of HCCP, so here we can focus on a single router in a simple network. We use the same network as above with the flux definitions from Table 2. In the earlier experiments we noted that the fluxes converged to a steady state after about 100 time units, so we do not begin to run HCCP until that point in the simulation. Thereafter we run the HCCP algorithm every 25 time units. In the first set of results we simply optimize throughput, while in the second we set $U_i(x_i) = w_i \log x_i$.

These studies bring us one step closer to evaluating the performance of HCCP in a real network. Our optimization framework takes offered loads as input and then optimizes link ranks and carried loads. The additional complexity we address in these simulations, and which was also included in those in Section 3, is that an endpoint application will typically change its offered load over time. In particular, greedy applications are likely to increase their offered load upon sensing that they are receiving less throughput than desired. This is exactly the type of behavior enacted by the greedy (GR) and near-greedy (NG) fluxes described in Table 1. We assume the endpoint applications receive immediate, accurate feedback on their allocated rates $(\hat{x}_I)$ and adjust their offered load in response at each time step.

4.1. Maximum throughput

Optimizing for maximum throughput is actually a degenerate case of the full HCCP method. The convex optimization step is reduced to enforcing the constraints presented as part of the tableau while hill climbing. To evaluate a new vector of ranks we calculate the resulting sum of allocated rates. Applying this approach to the experimental scenario detailed above we obtain the results in Fig. 7. The simulation is allowed to run for 100 time units before the optimal ranks are applied. At this point the total throughput begins to climb, showing a transient maximum of about 2950 compared to the previous 2250. The absolute maximum possible is 3000 which is the total capacity of the three output links.

The sawtooth shape of the curve is a composite result of the underlying behavior of the nine fluxes being carried by the router. The rational and near-rational fluxes gradually converge to their fair shares while the greedy and near-

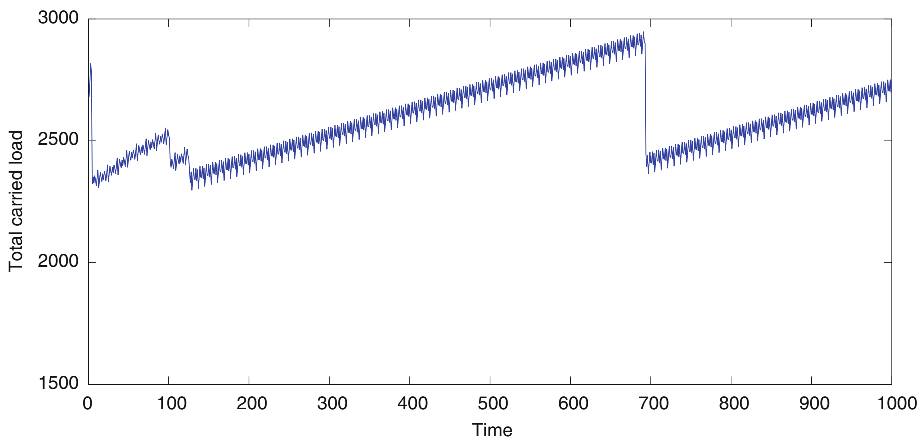

Fig. 7. Throughput maximization.

greedy fluxes persist in their efforts to escape regulation. The exact shape of the curve is a result of applying the fair maximum bandwidth function to the loads offered to each output link, and is bounded from above by the sum of the fair bandwidth allocations on each link, $\sum b_u^k$.

The major drawback in simply optimizing throughput is that the optimizer can push the rank of one or more links down to zero in its quest for an optimal solution. In this particular case, links 1 and 3 are given ranks of zero while link 2 is given a rank of 1. That is, only data incoming on link 2 would be forwarded. This is certainly in accord with the mathematical details of the problem formulation, but it is not something that either network operators or customers would expect to see. As a result we prefer a slightly different objective function.

4.2. Proportional fairness

We change the utility function to $U_i(\hat{x}_i) = \log \hat{x}_i$. This could be further generalized to $U_i(\hat{x}_i) = w_i \log \hat{x}_i$ but we do not introduce that extra complexity here. The additional flexibility of using weights w_i in the utility function could be used to adjust the network operator's relative preference for individual fluxes.

Repeating the same run as presented in Section 4.1 we obtain the results shown in Figs. 8 and 9. The final ranks for links 1, 2 and 3 are 0.395, 0.187 and 0.419, respectively. Proportionally fair allocation enables portions of the incoming fluxes on all three links to be forwarded.

The average total carried load decreases only slightly from 2590 when using throughput maximization to 2450 for proportionally fair allocation. Fig. 8 shows the carried traffic for flux 1, a flux with rational behavior and Fig. 9 shows the carried traffic for flux 3, a greedy flux. The rational flux is served quite well and converges fairly quickly to its fair maximum bandwidth. The greedy flux persists in trying to escape regulation but is held to a carried load at or below its fair share. This objective function delivers proportionally fair performance and does not have the drawback of potentially reducing some fluxes to zero.

5. Previous work

General and somewhat vague appeals to the end-to-end argument [37] are often used to explain why one or more proposed functions should not be implemented inside the network. However, Saltzer et al were simply arguing the case for careful consideration of the location of

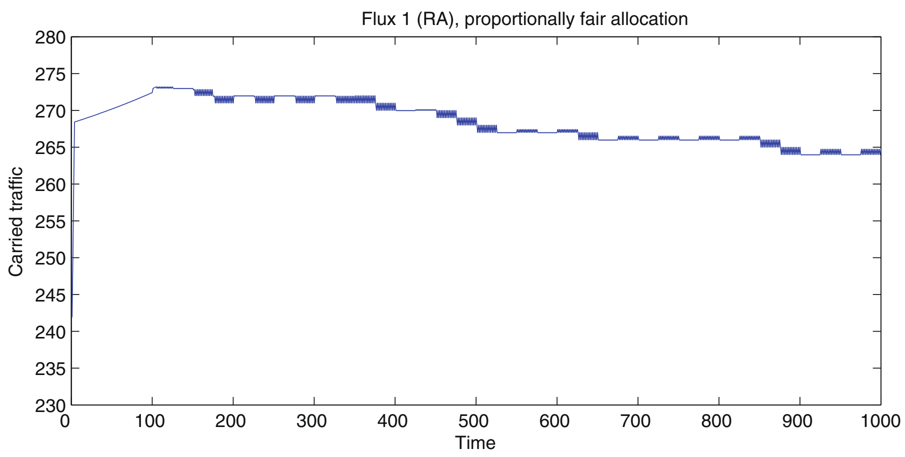

Fig. 8. Rational flux, proportional allocation.

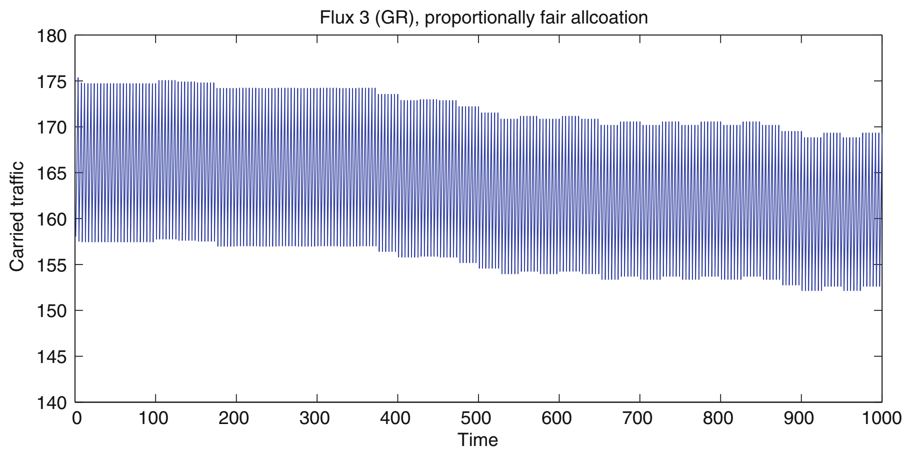

Fig. 9. Greedy flux, proportional allocation.

network services. They were not making the one-sided statement that only end-to-end control should be used. In particular they state:

> *"In a system that includes communications, one usually draws a modular boundary around the communication subsystem and defines a firm interface between it and the rest of the system. When doing so, it becomes apparent that there is a list of functions each of which might be implemented in any of several ways: by the communication subsystem, by its client, as a joint venture, or perhaps redundantly, each doing its own version."*

There are certainly a variety of ways in which bandwidth allocation and congestion control might be implemented and an extensive literature on those topics exists. It is interesting to note that the examples cited by Saltzer et al to motivate the end-to-end argument are focused on reliable, secure transmission. When it comes to network performance Saltzer et al allow that an analysis of the trade-offs can be very complex.

An implicit argument in their work is that the application layer often knows best whether a particular function is required, and if so, whether it should be invoked in a particular instance. This is certainly true when dealing with outcomes that concern a single user employing a single application. However, overall network performance, fair bandwidth allocation, and the onset of congestion are issues that go well past the boundaries of a single user or application. Mechanisms such as RED which require the network to signal an indication of congestion to the endpoints are relying on the decisions of individual users and applications. Where multiple greedy endpoints exist this simply becomes a path to a tragedy of the commons [15].

Nagle observed this in the ARPANET as early as 1984 [31] and stressed the effects of greedy behavior: *"Worth noting is that a few badly-behaved hosts can by themselves congest the gateways and prevent other hosts from passing traffic."* Both algorithms presented by Nagle, however, fall back to relying on endpoint control. His method for solving the small packet problem required changes to TCP, and ICMP source quenches rely on the cooperation of the sending TCP. It is interesting to note the circumstances at the time: *"The only serious problem comes from the User Datagram Protocol, not normally a major traffic generator."*

The closing section of Nagle's paper contains a suggestion based on observing the effects of a misbehaving host: *"We suggest that a worthwhile practical measure is to discard the latest packet from the host that originated the most packets currently queued within the gateway. This strategy will tend to balance throughput amongst the hosts using the gateway. We have not yet tried this strategy, but it seems a reasonable starting point for gateway self-protection.* This is very much along the lines of the mechanism we suggest. HCCP will balance the bandwidth allocation over fluxes based on a policy set by the domain administrator.

Jacobson [19] notes that a congestion avoidance mechanism which relies on endpoints requires two components: *"The network must be able to signal the transport endpoints that congestion is occurring (or about to occur). And the endpoints must have a policy that decreases utilization if this signal is received and increases utilization if the signal isn't received."* He then goes on to detail the mechanism added to TCP, and clearly the assumption of compliance by the endpoints is built in to this mechanism. The closing section of the paper is called *"Future work: the gateway side of congestion control"*. Jacobson envisioned using packet drops to signal compliant endpoint hosts and so concludes that noncompliant endpoints would implicitly be regulated by this same mechanism. The additional level of detail addressed by Jacobson, which we have not dealt with, is the actual measurement of bandwidth use. However, Jacobson anticipates the regulation of individual TCP sessions, whereas we advocate the control of aggregates. This may not remove the need to deal with burstiness when deriving measurements, especially if component flows of the fluxes are self-similar, but it does change the problem.

On the foundation built by researchers such as Clark, Jacobson, Jain, Nagle, Saltzer and many others a large body of work has been assembled which examines congestion control. At the time of writing there were over 500 citations to Jacobson's 1988 paper alone. The vast majority of the previous work is concerned with end-to-end control and falls into four classes: active queue management

(AQM) [7,8,18,27], explicit congestion notification (ECN) [22,23,32], variants of TCP [21,26,42], equation-based control [11,36], and "other" [45]. Of course the citations given here are meant only to be illustrative rather than exhaustive. More comprehensive surveys are available in [4,44].

The papers in the literature which are not limited to end-to-end control might be said to deal with "network modeling" more generally [41] and among these there are a significant number which deal specifically with the issue of greedy behavior [1,38].

The technique we propose in this paper is related to the ideas presented in [2,6,10,12,17,40]. Stoica et al. [40] extend fair queuing by having edge routers estimate the incoming rate of each flow and insert some indication of that estimate into each packet. Core routers then use probabilistic dropping based on the header information inserted at the edge plus a local estimate of aggregate flow. In an effort to ensure scalability no per-flow state is maintained in the core. They object to per-flow queuing partly on the basis that packet classification is required.

HCCP operates at a level and time scale similar to the fluid model presented by Stoica et al. They go further than we have here in establishing the technique required at finer time scales to perform arrival rate estimation. One contribution of HCCP in addition to CSFQ is the use of ranks propagated from endpoint hosts through the edge and into the core, and the use of these ranks to arrive at an optimal, globally fair allocation of bandwidth. The weights in weighted CSFQ have some resemblance to ranks but of course in Stoica's vision these weights were to be applied to individual flows. In HCCP the specification of fluxes rather than flows enhances scalability. It also prevents unfriendly flows from using various forms of subterfuge to escape regulation. In fact CSFQ is open to one of the forms of abuse we discuss earlier: in CSFQ all packets in a flow (as defined by Stoica et al.) must follow the same path through the core. The use of proxies is common in current "unfriendly" protocols and these would escape regulation by CSFQ. If implemented as part of the forwarding path HCCP can also overcome the objection that packet classification is required – this is the intrinsic nature of forwarding packets. They must be "classified" by a forwarding algorithm in order to select the best outbound interface. HCCP also incorporates the notion of punishment recognized by Stoica et al. However, further work is required to settle the buffer and queue management techniques required to support HCCP so that *fire-hose* applications do not still find some way to succeed by taking advantage of fluctuations at fine time scales.

HCCP also operates on aggregates rather than individual flows. Numerous efforts have been made to identify and classify flow aggregates mostly for three purposes: bandwidth policing [28,29], traffic modeling [3,39], and detecting and preventing network abuse (DoS attacks) [5,24]. All such previous attempts are based on guessing the combined properties of possibly unrelated flows that together appear to cause some kind of problem. None of them provides for a rigorous identification of the aggregate *user* of those flows, which, as we argue, is the indispensable prerequisite for a meaningful and foolproof implementation of the concept of fairness. Such an identification requires an extra mechanism: it cannot be based solely on the information available to the router within the IP packet.

The work by Floyd and Fall [10] is almost prescient in its anticipation of today's situation where we face "potential negative impacts from an increasing deployment of non-congestion-controlled best-effort traffic". However, the present work diverges quite quickly from their approach as Floyd and Fall immediately turn to a consideration of regulating individual flows. The flows to be regulated are identified on the basis of their drop history, which of course requires the maintenance of per-flow state. It also opens their technique up to evasion through the same sorts of subterfuge effective against CSFQ.

Garg et al. [12] also consider the regulation of individual flows and the maintenance of per-flow state to prevent congestion collapse due to selfish competition. They start with a fluid flow model and show the equilibria which result from the use of a variety of scheduling functions. The bandwidth allocation function which we have explored for use with HCCP is more aggressive than the RIS function of Garg et al. in that a flux which is sufficiently greedy will see its throughput drop to zero rather than approaching a positive asymptote as with RIS. Another significant difference is that HCCP arrives at a set of weights which optimizes network utility. DWS scheduling leaves its weights unspecified, and the only utility examined is that of the individual user. With HCCP we are proposing that overall network utility be managed by the network operator rather than leaving the position of the equilibrium up to the strategies of individual, possibly greedy, endpoints.

Albuquerque et al. [2] bring the control problem from the endpoints into the network edge in their "network border patrol" (NBP) and then propose the use of an extended version of CSFQ in the core. NBP is again a flow-level technique. It has the extra complexity that it attempts to maintain a balance on the number of packets of each flow in the network. This requires communication between the ingress and egress routers for each flow. The communication scheme is reminiscent of BECN, but operating between ingress and egress routers rather than endpoints and with backward congestion notifications being triggered by the reception of control rather than data packets. The results presented in [2] are very good but only cases with a few tens of flows are tested, and all flows persist for the full duration of each simulation.

He et al. [17] present an optimization approach similar in spirit to HCCP, but in a framework which tries to jointly optimize user and network operator utility. Our goal with HCCP is somewhat more general: we combine the flux ranks indicated by the end systems with the policies established by the network administrator and then optimize. In addition, we envision regulation of aggregates rather than individual users to keep scaling effects in check. The final heuristic presented by He et al., TRUMP, has been proven stable only at light loads whereas HCCP is asymptotically stable. HCCP, however, is more complex than TRUMP and the further work completed by He et al to establish the behavior of TRUMP at finer time scales is still future work for HCCP.

6. Conclusions

In comparison to previous work, the contributions of HCCP are:

- aggregates rather than individual flows or user sessions are the entities controlled,
- control is applied independently of protocol: HCCP is not just one more version of TCP,
- control resides within the network rather than relying on compliant endpoints,
- HCCP incorporates a mechanism by which endpoint hosts can indicate their preferences,
- these preferences are then modified by the ranking policy implemented at each router,
- there is no assumption that all packets within a flux follow the same path through the network, and
- greedy fluxes are penalized rather than just being regulated: they may receive zero bandwidth if they are sufficiently greedy.

Our numerical results show that HCCP is effective at regulating a wide range of rather generally characterized transport protocols, even when these are mixed on a single link. Our so-called "rational" fluxes achieve their fair shares even in the presence of a number of very aggressive, persistently greedy fluxes. Bandwidth is allocated optimally amongst fluxes via the combination of hill climbing and convex programming. A variety of objective functions can be used with this technique, but we deprecate the use of throughput maximization because it can easily lead to the situation where one or more fluxes receive zero bandwidth. Proportionally fair allocation is one possibility presented here that seems to perform better.

HCCP is a general framework for implementing fair operation of a potentially large network catering to diverse collections of bandwidth-hungry applications. Our scheme does not try to interfere with the fundamental connectionless paradigm of internetworking: its underlying assumption is that all traffic essentially consists of abstract streams of packets (called fluxes) whose organization follows clear administrative rules. The definition of a flux captures the relevant attributes of a group of packets from the viewpoint of their fair treatment at a router without assuming any transport-layer membership of that group. The router's policy with respect to fluxes kicks in at the time of congestion and deals with those fluxes whose offered load exceeds their fair share. Such fluxes are penalized in a way that effectively enforces their social responsiveness to congestion. Social behavior is in the flux's best interest as it is the only way the flux can maximize its quality of service.

The scope of a flux and the identity of its *owner* (i.e. the entity to which the flux is attributed for the purpose of enforcing its fair behavior) depend on the place in the network. For example the different users of a single host can be viewed by the host administrator as contributors of different fluxes (all packets created by the same user belong to one flux), while all the packets contributed by that host can be treated as a single flux by its edge router. Notably this hierarchy need not be strictly enforced at absolutely all levels. In particular, it is conceivable that a host defines no flux policy at all while its edge router does. In such a case the router will still provide for a fair treatment of that particular host (viewed as one of its *users*) while the users of that host may experience unfair service. This kind of unfairness is local: it results from greediness within the host rather than unfair treatment of the host by the network. One contribution of our work is a way to build administrative hierarchies of ranks to meaningfully aggregate fairness policies of hosts and routers. On top of those hierarchies we propose a generic mechanism for maximizing a router's performance in a way that preserves the fair treatment of its fluxes. This local optimization when followed by all routers translates into a globally fair network.

Our hierarchy of fluxes can also be viewed as a way to equip the connectionless network with the minimum characteristics of a connection-oriented system to provide for (fair) "bandwidth guarantees". Needless to say, the "standard" approach consisting of setting aside some bandwidth for those transport-layer sessions that require bandwidth guarantees is not guaranteed to yield fairness. Moreover, it will bring about bandwidth fragmentation and connection delays which are both avoided in our approach. Note that it is still possible within our framework to have absolutely hard bandwidth guarantees by declaring a highly ranked flux encompassing a single transport-layer session. As the mechanism of conveying flux ranks among routers is dynamic, the rigid connection-oriented approach to quality of service becomes a trivial subset of our scheme, which, at its full power, is incomparably more flexible.

References

[1] A. Akella, S. Seshan, R. Karp, S. Shenker, C. Papadimitriou, Selfish behavior and stability of the Internet: a game-theoretic analysis of TCP, SIGCOMM Comput. Commun. Rev. 32 (4) (2002) 117–130.

[2] C. Albuquerque, B. Vickers, T. Suda, Network border patrol: preventing congestion collapse and promoting fairness in the Internet, IEEE/ACM Trans. Netw. 12 (1) (2004) 173–186.

[3] K. Chan Lan, J. Heidemann, A measurement study of correlations of internet flow characteristics, Comput. Netw. 50 (1) (2006) 46–62.

[4] G. Chatranon, M. Labrador, S. Banerjee, A survey of TCP-friendly router-based AQM schemes, Comput. Commun. 27 (15) (2004) 1424–1440. Sept. 22.

[5] S. Chen, Y. Tang, W. Du, Stateful DDoS attacks and targeted filtering, J. Netw. Comput. Appl. 30 (3) (2007) 823–840.

[6] M. Chiang, J. Lee, R. Calderbank, D. Palomar, M. Fazel, Network utility maximization with nonconcave coupled and reliability-based utilities, SIGMETRICS Perform. Eval. Rev. 33 (1) (2005) 277.

[7] S. Deb, R. Srikant, Rate-based versus queue-based models of congestion control, in: SIGMETRICS'04/Performance'04: Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, ACM, 2004, pp. 246–257.

[8] A. Demers, S. Keshav, S. Shenker, Analysis and simulation of a fair queueing algorithm, SIGCOMM Comput. Commun. Rev. 19 (4) (1989) 1–12.

[9] S. Floyd, V. Jacobson, Link-sharing and resource management models for packet networks, IEEE/ACM Trans. Netw. 3 (4) (1995) 365–386.

[10] S. Floyd, K. Fall, Router mechanisms to support end-to-end congestion control, Technical Report, Lawrence Berkeley National Laboratory, 1997.

[11] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-based congestion control for unicast applications, in: SIGCOMM'00: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, New York, NY, USA, ACM, 2000, pp. 43–56.

[12] R. Garg, A. Kamra, V. Khurana, A game-theoretic approach towards congestion control in communication networks, SIGCOMM Comput. Commun. Rev. 32 (3) (2002) 47–61.
[13] V. Govindaswamy, G. Zaruba, G. Balasekaran, RECHOKe: a scheme for detection, control and punishment of malicious flows in IP networks, in: Proceedings of GLOBECOM'07, November 2007, pp. 12–21.
[14] Y. Gu, R.L. Grossman, UDT: UDP-based data transfer for high-speed wide area networks, Comput. Netw. 51 (7) (2007) 1777–1799.
[15] G. Hardin, The tragedy of the commons, Science 162 (3859) (1968) 1243–1248.
[16] E. He, J. Leigh, O. Yu, T.A. DeFanti, Reliable blast UDP: predictable high performance bulk data transfer, in: CLUSTER'02: Proceedings of the IEEE International Conference on Cluster Computing, Washington, DC, USA, IEEE Computer Society, 2002, p. 317.
[17] J. He, M. Suchara, M. Bresler, J. Rexford, M. Chiang, Rethinking Internet traffic management: from multiple decompositions to a practical protocol, in: CoNEXT'07: Proceedings of the 2007 ACM CoNEXT Conference, New York, NY, USA, ACM, 2007, pp. 1–12.
[18] G. Iannaccone, M. May, C. Diot, Aggregate traffic performance with active queue management and drop from tail, SIGCOMM Comput. Commun. Rev. 31 (3) (2001) 4–13.
[19] V. Jacobson, Congestion avoidance and control, in: SIGCOMM'88: Symposium Proceedings on Communications Architectures and Protocols, New York, NY, USA, ACM, 1988, pp. 314–329.
[20] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, Inferring TCP connection characteristics through passive measurements, in: INFOCOM 2004, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, 2004, pp. 1582–1592.
[21] S. Jin, L. Guo, I. Matta, A. Bestavros, A spectrum of TCP-friendly window-based congestion control algorithms, IEEE/ACM Trans. Netw. 11 (3) (2003) 341–355.
[22] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, SIGCOMM Comput. Commun. Rev. 32 (4) (2002) 89–102.
[23] F. Kelly, G. Raina, T. Voice, Stability and fairness of explicit congestion control with small buffers, SIGCOMM Comput. Commun. Rev. 38 (3) (2008) 51–62.
[24] S.S. Kim, A.L.N. Reddy, Statistical techniques for detecting traffic anomalies through packet header data, IEEE/ACM Trans. Netw. 16 (3) (2008) 562–575.
[25] W.-C. Liao, F. Papadopoulos, K. Psounis, Performance analysis of BitTorrent-like systems with heterogeneous users, Perform. Eval. 64 (9–12) (2007) 876–891.
[26] S. Low, L. Peterson, L. Wang, Understanding TCP Vegas: a duality model, in: SIGMETRICS'01: Proceedings of the 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, ACM, 2001, pp. 226–235.
[27] Q. Ma, K. Ramakrishnan, Queue management for explicit rate based congestion control, in: SIGMETRICS'97: Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, ACM, 1997, pp. 39–51.
[28] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, S. Shenker, Controlling high bandwidth aggregates in the network, SIGCOMM Comput. Commun. Rev. 32 (3) (2002) 62–73.
[29] R. Mahajan, S. Floyd, D. Wetherall, Controlling high-bandwidth flows at the congested router, in: ICNP, 2001, p. 0192.
[30] J. Moy. RFC 2328: OSPF version 2, See also STD0054, Obsoletes RFC2178, Status: STANDARD, April 1998.
[31] J. Nagle, Congestion control in IP/TCP internetworks, SIGCOMM Comput. Commun. Rev. 14 (4) (1984) 11–17.
[32] F. Paganini, Z. Wang, J. Doyle, S. Low, Congestion control for high performance stability and fairness in general networks, IEEE/ACM Trans. Netw. 13 (1) (2005) 43–56.
[33] D. Qiu, R. Srikant, Modeling and performance analysis of BitTorrent-like peer-to-peer networks, SIGCOMM Comput. Commun. Rev. 34 (4) (2004) 367–378.
[34] Y. Qu, J. Luo, W. Li, B. Liu, L.T. Yang, Square: a new TCP variant for future high speed and long delay environments, in: AINA, 2008, pp. 636–643.
[35] Y. Rekhter, T. Li, RFC 1771: A Border Gateway Protocol 4 (BGP-4), Obsoletes RFC1654, Status: DRAFT STANDARD, March 1995.
[36] I. Rhee, L. Xu, Limitations of equation-based congestion control, IEEE/ACM Trans. Netw. 15 (4) (2007) 852–865.
[37] J.H. Saltzer, D.P. Reed, D.D. Clark, End-to-end arguments in system design, ACM Trans. Comput. Syst. 2 (4) (1984) 277–288.
[38] S. Shenker, Making greed work in networks: a game-theoretic analysis of switch service disciplines, in: SIGCOMM'94: Proceedings of the Conference on Communications Architectures, Protocols and Applications, New York, NY, USA, ACM, 1994, pp. 47–57.
[39] W. Shi, M. MacGregor, P. Gburzynski, Synthetic trace generation for the Internet: an integrated model, in: SPECTS 2004: Symposium on Performance Evaluation of Computer and Telecommunication Systems, Society for Computer Simulation, 2004, pp. 471–477.
[40] I. Stoica, S. Shenker, H. Zhang, Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high speed networks, SIGCOMM Comput. Commun. Rev. 28 (4) (1998) 118–130.
[41] T. Voice, A global stability result for primal–dual congestion control algorithms with routing, SIGCOMM Comput. Commun. Rev. 34 (3) (2004) 35–41.
[42] Z. Wang, J. Crowcroft, Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm, SIGCOMM Comput. Commun. Rev. 22 (2) (1992) 9–16.
[43] P. White, RSVP and integrated services in the Internet: a tutorial, IEEE Commun. Mag. 35 (5) (1997) 100–106.
[44] J. Widmer, R. Denda, M. Mauve, A survey on TCP-friendly congestion control, IEEE Netw. 15 (4) (2001) 28–72. Jul-Aug.
[45] C. Williamson, Optimizing file transfer response time using the loss-load curve congestion control mechanism, SIGCOMM Comput. Commun. Rev. 23 (4) (1993) 117–126.

Pawel Gburzynski received his MSc and PhD in Computer Science from the University of Warsaw, Poland in 1976 and 1982, respectively. Before coming to Canada in 1984, he had been a research associate, systems programmer, and consultant in the Department of Mathematics, Informatics, and Mechanics at the University of Warsaw. Since 1985, he has been with the Department of Computing Science, University of Alberta, where he is a Professor. His research interests are in communication networks, embedded systems, operating systems, simulation, and performance evaluation.

M.H. MacGregor is Professor and Chair of the Department of Computing Science at the University of Alberta. His current research interests are in architectures and algorithms for routing and switching. He received a PhD in Computing Science from the University of Alberta in 1991 for his dissertation on self traffic-engineering networks, the research for which was conducted at TRLabs. He has been a Registered Professional Engineer in the Province of Alberta since 1980, and has over ten years of industrial experience in the areas of real-time process control, broadband Internet service architectures, and optical switching. He is a Senior Member of IEEE and a member of ACM, APEGGA and IEICE.