Network Border Patrol

Célio Albuquerque[†], Brett J. Vickers[‡] and Tatsuya Suda[†]

† Dept. of Information and Computer Science
University of California, Irvine
{celio,suda}@ics.uci.edu

‡ Dept. of Computer Science
Rutgers University
bvickers@cs.rutgers.edu

Abstract— **The end-to-end nature of Internet congestion control is an important factor in its scalability and robustness. However, end-to-end congestion control algorithms alone are incapable of preventing the congestion collapse and unfair bandwidth allocations created by applications which are unresponsive to network congestion. In this paper, we propose and investigate a new congestion avoidance mechanism called** *Network Border Patrol* **(NBP). NBP relies on the exchange of feedback between routers at the borders of a network in order to detect and restrict unresponsive traffic flows before they enter the network. The NBP mechanism is compliant with the Internet philosophy of pushing complexity toward the edges of the network whenever possible. Simulation results show that NBP effectively eliminates congestion collapse, and that, when combined with fair queueing, NBP achieves approximately max-min fair bandwidth allocations for competing network flows.**

Keywords—**Internet, congestion control, congestion collapse, max-min fairness, end-to-end argument**

I. Introduction

THE essential philosophy behind the Internet is expressed by the scalability argument: no protocol, algorithm or service should be introduced into the Internet if it does not scale well. A key corollary to the scalability argument is the end-to-end argument: to maintain scalability, algorithmic complexity should be pushed to the edges of the network whenever possible. Perhaps the best example of the Internet philosophy is TCP congestion control, which is achieved primarily through algorithms implemented at end systems. Unfortunately, TCP congestion control also illustrates some of the shortcomings of the end-to-end argument.

As a result of its strict adherence to end-to-end congestion control, the current Internet suffers from two maladies: congestion collapse from undelivered packets, and unfair allocations of bandwidth between competing traffic flows. The first malady—congestion collapse from undelivered packets—arises when bandwidth is continuously consumed by packets that are dropped before reaching their ultimate destinations [1]. Unresponsive flows,[1] which are becoming increasingly prevalent in the Internet as network applications using audio and video become more popular, are the primary cause of this type of congestion collapse, and the Internet currently has no way of effectively regulating them.

This research is supported by the National Science Foundation through grant NCR-9628109. It has also been supported by grants from the University of California MICRO program, Hitachi America, Hitachi, Standard Microsystem Corp., Canon Information Systems Inc., Nippon Telegraph and Telephone Corp. (NTT), Nippon Steel Information and Communica-tion Systems Inc. (ENICOM), Tokyo Electric Power Co., Fujitsu, Novell, Matsushita Electric Industrial Co. and Fundação CAPES/Brazil.

[1] An unresponsive flow is any flow generated by an application that fails to reduce its transmission rate in response to increased packet discarding caused by congestion.

The second malady—unfair bandwidth allocation—arises in the Internet for a variety of reasons, one of which is the presence of unresponsive flows. Adaptive flows (e.g., TCP flows) that respond to congestion by rapidly reducing their transmission rates are likely to receive unfairly small bandwidth allocations when competing with unresponsive or malicious flows. The Internet protocols themselves also introduce unfairness. The TCP algorithm, for instance, inherently causes each TCP flow to receive a bandwidth that is inversely proportional to its round trip time [2]. Hence, TCP connections with short round trip times may receive unfairly large allocations of network bandwidth when compared to connections with longer round trip times.

These maladies—congestion collapse from undelivered packets and unfair bandwidth allocations—have not gone unrecognized. Some have argued that they may be mitigated through the use of improved packet scheduling [3] or queue management [4] mechanisms in network routers. For instance, per-flow packet scheduling mechanisms like Weighted Fair Queueing (WFQ) [5], [6] attempt to offer fair allocations of bandwidth to flows contending for the same link. So does Core-Stateless Fair Queueing (CSFQ) [7], an approximation of WFQ that requires only edge routers to maintain per-flow state. Active queue management mechanisms like Fair Random Early Detection (FRED) [8] achieve an effect similar to fair queueing by discarding packets from flows that are using more than their fair share of a link's bandwidth. All of these mechanisms are more complex and expensive to implement than simple FIFO queueing, but they reduce the causes of unfairness and congestion collapse in the Internet. Nevertheless, they do not eradicate them. For illustration of this fact, consider the example shown in Figure 1. In this example, two unresponsive flows compete for bandwidth in a network containing two bottleneck links arbitrated by a fair queueing mechanism. At the first bottleneck link (R_1-R_2), fair queueing ensures that each flow receives half of the link's available bandwidth (750 kbps). On the second bottleneck link (R_2-S_4), much of the traffic from flow B is discarded due to the link's limited capacity (128 kbps). Hence, flow A achieves a throughput of 750 kbps and flow B achieves a throughput of 128 kbps. Clearly, congestion collapse has occurred, because flow B packets, which are ultimately discarded on the second bottleneck link, unnecessarily limit the throughput of flow A across the first bottleneck link. Furthermore, while both flows receive equal bandwidth allocations on the first bottleneck link, their allocations are not *globally max-min fair*.[2] A globally max-min fair alloca-

[2] An allocation of bandwidth is said to be globally max-min fair if, at every link,

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2000**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **Network Border Patrol**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Information and Computer Science, University of California, Irvine**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**10**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

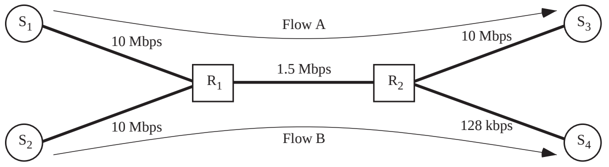


Fig. 1. Example of a network which experiences congestion collapse

tion of bandwidth would have been 1.372 Mbps for flow A and 128 kbps for flow B.

This example, which is a variant of an example presented in [1], illustrates the inability of local scheduling mechanisms, such as WFQ, to eliminate congestion collapse and achieve global max-min fairness without the assistance of additional network mechanisms.

Jain *et al.* have proposed several rate control algorithms that are able to prevent congestion collapse and provide global max-min fairness to competing flows [10]. These algorithms (e.g., ERICA, ERICA+) are designed for the ATM Available Bit Rate (ABR) service and require all network switches to compute fair allocations of bandwidth among competing connections. However, these algorithms are not easily tailorable to the current Internet, because they violate the Internet design philosophy of keeping router implementations simple and pushing complexity to the edges of the network.

Floyd and Fall have approached the problem of congestion collapse by proposing low-complexity router mechanisms that promote the use of adaptive or "TCP-friendly" end-to-end congestion control [1]. Their suggested approach requires selected gateway routers to monitor high-bandwidth flows in order to determine whether they are responsive to congestion. Flows that are determined to be unresponsive are penalized by a higher packet discarding rate at the gateway router. A limitation of this approach is that the procedures currently available to identify unresponsive flows are not always successful [7].

In this paper, we introduce and investigate a new Internet traffic control mechanism called *Network Border Patrol* (NBP). The basic principle of NBP is to compare, at the borders of the network, the rates at which each flow's packets are entering and leaving the network. If packets are entering the network faster than they are leaving it, then the network is very likely to be buffering or, worse yet, discarding the flow's packets. In other words, the network is receiving more packets than it can handle. NBP prevents this scenario by "patrolling" the network's borders, ensuring that packets do not enter the network at a rate greater than they are able to leave it. This has the beneficial effect of preventing congestion collapse from undelivered packets, because an unresponsive flow's otherwise undeliverable packets never enter the network in the first place.

NBP's prevention of congestion collapse comes at the expense of some additional network complexity, since routers at the borders of the network (i.e., edge routers) are expected to monitor and control the rates of individual flows. NBP also introduces an added communication overhead, since in order for an edge router to know the rate at which its packets are leaving the network, it must exchange feedback with other edge routers. However, unlike other existing approaches to the problem of congestion collapse, NBP's added complexity is isolated to edge routers; routers within the core of the network remain unchanged. Moreover, end systems operate in total ignorance of the fact that NBP is implemented in the network, so no changes to transport protocols are necessary.

all active flows not bottlenecked at another link are allocated a maximum, equal share of the link's remaining bandwidth [9].

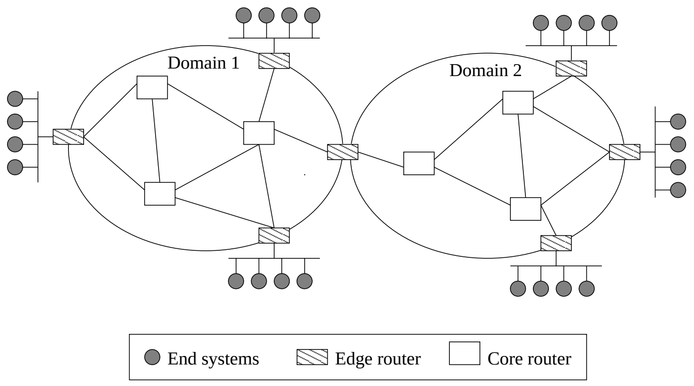


Fig. 2. The core-stateless Internet architecture assumed by NBP

Note that the primary goal of NBP is to prevent congestion collapse from undelivered packets. On its own, NBP cannot provide global max-min fairness to competing network flows. Nevertheless, when combined with fair queueing at core routers, NBP can achieve approximate global max-min fairness, as we will show later in this paper.

The remainder of this paper is organized as follows. In section II, we describe the architectural components of the Network Border Patrol mechanism in further detail and present the feedback and rate control algorithms used by NBP edge routers to prevent congestion collapse. In section III, we present the results of several simulations, which illustrate the ability of NBP to avoid congestion collapse and, when combined with a fair queueing algorithm in core routers, to provide global max-min fairness to competing network flows. In section IV, we discuss several implementation and scalability issues that must be addressed in order to make deployment of NBP feasible in the Internet. Finally, in section V we provide some concluding remarks.

II. Network Border Patrol

Network Border Patrol is a core-stateless congestion avoidance mechanism. That is, it is aligned with the core-stateless approach [7], which allows routers on the borders (or edges) of a network to perform flow classification and maintain per-flow state but does not allow routers at the core of the network to do so. Figure 2 illustrates this architecture. In this paper, we draw a further distinction between two types of edge routers. Depending on which flow it is operating on, an edge router may be viewed as an *ingress* or an *egress* router. An edge router operating on a flow passing into a network is called an ingress router, whereas an edge router operating on a flow passing out of a network is called

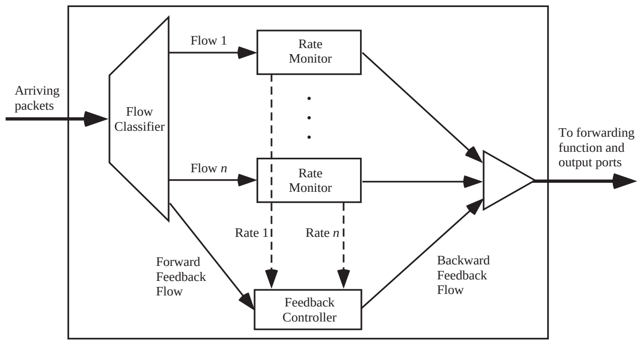


Fig. 3. An input port of an NBP egress router

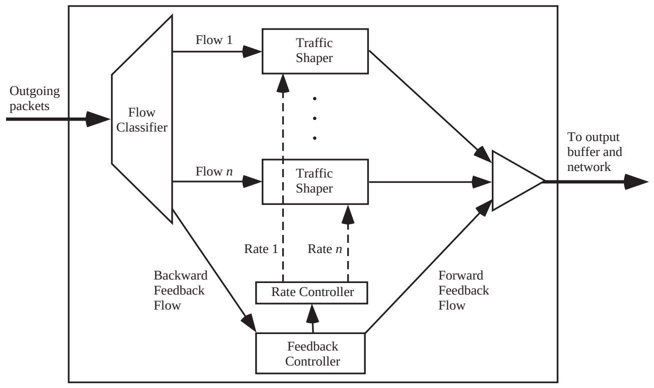


Fig. 4. An output port of an NBP ingress router

an egress router. Note that a flow may pass through more than one egress (or ingress) router if the end-to-end path crosses multiple networks.

NBP prevents congestion collapse through a combination of per-flow rate monitoring at egress routers and per-flow rate control at ingress routers. Rate monitoring allows an egress router to determine how rapidly each flow's packets are leaving the network, whereas rate control allows an ingress router to police the rate at which each flow's packets enter the network. Linking these two functions together are the feedback packets exchanged between ingress and egress routers; ingress routers send egress routers *forward* feedback packets to inform them about the flows that are being rate controlled, and egress routers send ingress routers *backward* feedback packets to inform them about the rates at which each flow's packets are leaving the network.

This section describes three important aspects of the NBP mechanism: (1) the architectural components, namely the modified edge routers, which must be present in the network, (2) the feedback control algorithm, which determines how and when information is exchanged between edge routers, and (3) the rate control algorithm, which uses the information carried in feedback packets to regulate flow transmission rates and thereby prevent congestion collapse in the network.

A. Architectural Components

The only components of the network that require modification by NBP are edge routers. The input ports of egress routers must be modified to perform per-flow monitoring of bit rates, and the output ports of ingress routers must be modified to perform per-flow rate control. In addition, both the ingress and the egress routers must be modified to exchange and handle feedback.

Figure 3 illustrates the architecture of an NBP egress router's input port. Packets sent by ingress routers arrive at the input port of the egress router and are first classified by flow. In the case of IPv6, this is done by examining the packet header's flow label, whereas in the case of IPv4, it is done by examining the packet's source and destination addresses and port numbers. Each flow's bit rate is then rate monitored using a rate estimation algorithm such as the Time Sliding Window (TSW) [11]. These rates are collected by a feedback controller, which returns them in backward feedback packets to an ingress router whenever a forward feedback packet arrives from that ingress router. In some cases, to be described later in this section, backward feedback packets are also generated asynchronously; that is, an egress router sends them to an ingress router without first waiting for a forward feedback packet.

The output ports of NBP ingress routers are also enhanced. Each contains a flow classifier, per-flow traffic shapers (e.g., leaky buckets), a feedback controller, and a rate controller. See Figure 4. The flow classifier classifies packets into flows, and the traffic shapers limit the rates at which packets from individual flows enter the network. The feedback controller receives backward feedback packets returning from egress routers and passes their contents to the rate controller. It also generates forward feedback packets, which it periodically transmits to the network's egress routers. The rate controller adjusts traffic shaper parameters according to a TCP-like rate control algorithm, which is described later in this section.

B. The Feedback Control Algorithm

The NBP feedback control algorithm determines how and when feedback packets are exchanged between edge routers. Feedback packets take the form of ICMP packets and are necessary in NBP for three reasons. First, they allow egress routers to discover which ingress routers are acting as sources for each of the flows they are monitoring. Second, they allow egress routers to communicate per-flow bit rates to ingress routers. Third, they allow ingress routers to detect network congestion and control their feedback generation intervals by estimating edge-to-edge round trip times.

The contents of NBP feedback packets are shown in Figure 5. Contained within the forward feedback packet is a time stamp and a list of flow specifications[3] for flows originating at the ingress router. The time stamp is used to calculate the round trip time between two edge routers, and the list of flow specifications indicates to an egress router the identities of active flows originating at the ingress router. (An edge router adds a flow to its list of active flows whenever a packet from a new flow arrives; it removes a flow when the flow becomes inactive.) In the event that the net-

[3] A flow specification is a value uniquely identifying a flow. In IPv6 it is the flow's flow label. In IPv4, it is the combination of source address, destination address, source port number, and destination port number.

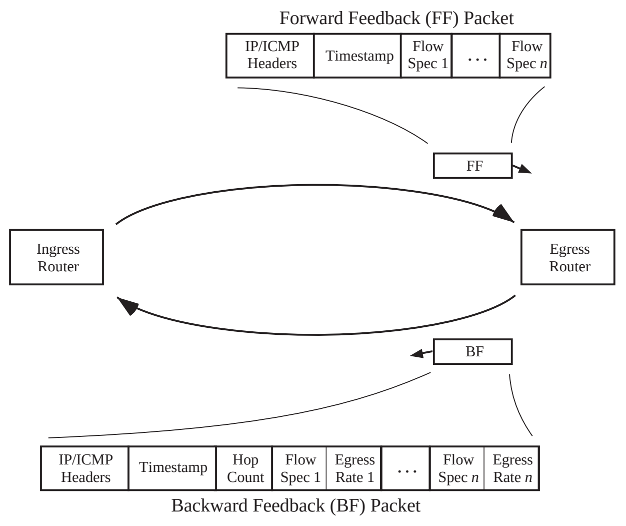


Fig. 5. Forward and backward feedback packets exchanged by edge routers

work's maximum transmission unit size is not sufficient to hold an entire list of flow specifications, multiple forward feedback packets are used.

When an egress router receives a forward feedback packet, it immediately generates a backward feedback packet and returns it to the ingress router. Contained within the backward feedback packet are the forward feedback packet's original time stamp, a router hop count, and a list of observed bit rates, called *egress rates*, collected by the egress router for each flow listed in the forward feedback packet. The router hop count, which is used by the ingress router's rate control algorithm, indicates how many routers are in the path between the ingress and the egress router. The egress router determines the hop count by examining the time to live (TTL) field of arriving forward feedback packets. When the backward feedback packet arrives at the ingress router, its contents are passed to the ingress router's rate controller, which uses them to adjust the parameters of each flow's traffic shaper.

In order to determine how often to generate forward feedback packets, an ingress router keeps, for each egress router, a timer which determines the frequency of forward feedback packet generation. To maintain an adequate and consistent feedback update interval, the timer repeatedly expires after an interval of time known as the *base round trip time*. The base round trip time for egress router *e*, denoted *e.baseRTT*, is defined as the shortest observed round trip time between the ingress router and egress router *e*, and it generally reflects the round trip time between the two edge routers when the network is not congested. The value *e.baseRTT* is calculated by estimating the current round trip time from each arriving backward feedback packet and updating *e.baseRTT* whenever the current round trip time is less.

Egress routers may also generate backward feedback packets asynchronously. If an egress router does not receive a forward feedback packet from an ingress router within a fixed interval of time (denoted *AsynchInterval*), it generates and transmits a backward feedback packet to the ingress router. Asynchronously generated backward feedback packets are specially marked by the

```
on arrival of BF packet p from egress router e
  if (p.asynchronous == FALSE)
    e.currentRTT = cur_time - p.timestamp;
    if (e.currentRTT < e.baseRTT)
      e.baseRTT = e.currentRTT;
    deltaRTT = e.currentRTT - e.baseRTT;
    for each flow f listed in p
      f.mrc = min (MSS / e.currentRTT, f.egress_rate / MF);
      if (f.phase == SLOW_START)
        if ($deltaRTT \times f.ingress\_rate < MSS \times e.hopcount$)
          $f.ingress\_rate = f.ingress\_rate \times 2$;
        else
          f.phase = CONG_AVOID;
      if (f.phase == CONG_AVOID)
        if ($deltaRTT \times f.ingress\_rate < MSS \times e.hopcount$)
          $f.ingress\_rate = f.ingress\_rate + f.mrc$;
        else
          f.ingress_rate = f.egress_rate - f.mrc;
  else /* p.asynchronous == TRUE */
    for each flow f listed in p
      if (f.phase == SLOW_START)
        if ($f.ingress\_rate > f.egress\_rate \times 8$)
          f.ingress_rate = f.egress_rate - f.mrc;
          f.phase = CONG_AVOID;
      else /* f.phase == CONG_AVOID */
        if ($f.ingress\_rate > f.egress\_rate + 3 \times f.mrc$)
          f.ingress_rate = f.egress_rate - f.mrc;
```

Fig. 6. Pseudocode for ingress router rate control algorithm

egress router and are not used by the ingress router to update the round trip time measurement. The reason for asynchronous backward feedback packet generation is to prevent the squelching of congestion feedback when forward feedback packets are delayed or dropped by the network. It also ensures that ingress routers receive frequent rate feedback and are able to respond to congestion even when the distance between edge routers is very large.

C. *The Rate Control Algorithm*

The NBP rate control algorithm regulates the rate at which each flow enters the network. Its primary goal is to converge on a set of per-flow transmission rates (hereinafter called *ingress rates*) that prevents congestion collapse from undelivered packets. It also attempts to lead the network to a state of maximum link utilization and low router buffer occupancies, and it does this in a manner that is similar to TCP.

In the NBP rate control algorithm, shown in Figure 6, a flow may be in one of two phases, *slow start* or *congestion avoidance*, which are similar to the phases of TCP congestion control. New flows enter the network in the slow start phase and proceed to the congestion avoidance phase only after the flow has experienced congestion. The rate control algorithm is invoked whenever a backward feedback (BF) packet arrives at an ingress router. Recall that egress routers send two types of BF packets to ingress routers: normal BF packets, which are generated when an egress router receives a forward feedback (FF) packet, and asynchronous BF packets, which egress routers generate without any prompting from an ingress router. Both types of BF packets contain a list of flows arriving at the egress router from the ingress router as well as the monitored egress rate for each flow. However, only normal BF packets contain meaningful time stamps which are copied from arriving FF packets.

If the arriving BF packet is a normal BF packet, then the algorithm calculates the current round trip time and updates the base round trip time, if necessary. It then calculates *deltaRTT*, which is the difference between the current round trip time (*e.currentRTT*) and the base round trip time (*e.baseRTT*). A *deltaRTT* value greater than zero indicates that packets are requiring a longer time to traverse the network than they once did, and this can only be due to the buffering of packets within the network.

NBP's rate control algorithm decides that a flow is experiencing congestion whenever it estimates that the network has buffered the equivalent of more than one of the flow's packets at each router hop. To do this, the algorithm first computes the product of the flow's ingress rate and *deltaRTT*. This value provides an estimate of the amount of flow data that is buffered somewhere in the network. If it is greater than the number of router hops between the ingress and the egress router multiplied by the size of the largest possible packet, then the flow is considered to be experiencing congestion. The rationale for determining congestion in this way is to maintain both high link utilization and low queueing delay. Ensuring there is always at least one packet buffered for transmission on a network link is the simplest way to achieve full utilization of the link, and deciding that congestion exists when more than one packet is buffered at the link keeps queueing delays low.

When the rate control algorithm determines that a flow is not experiencing congestion, it increases the flow's ingress rate. If the flow is in the slow start phase, its ingress rate is doubled. Doubling the ingress rate allows a new flow to rapidly capture available bandwidth if the network is underutilized. If the flow is in the congestion avoidance phase, its ingress rate is conservatively incremented by a *minimum rate change* (MRC) value in order to avoid the creation of congestion. MRC is computed as the maximum segment size divided by the current round trip time between the edge routers. This results in rate growth behavior that is similar to TCP in its congestion avoidance phase. Furthermore, MRC is not allowed to exceed the flow's current egress rate divided by a constant factor (MF). This guarantees that rate increments are not excessively large when the round trip time is small.

When the rate control algorithm determines that a flow is experiencing congestion, it reduces the flow's ingress rate. If a flow is in the slow start phase, it enters the congestion avoidance phase. If a flow is already in the congestion avoidance phase, its ingress rate is reduced to the flow's egress rate decremented by MRC. In other words, an observation of congestion forces the ingress router to send the flow's packets into the network at a rate slightly lower than the rate at which they are leaving the network.

The actions described above are taken only when a normal BF packet arrives at an ingress router. A different set of actions is taken when an asynchronous BF packet arrives. This is because, unlike normal BF packets, asynchronous BF packets are not generated in response to FF packets and thus do not carry meaningful time stamps. Therefore, the congestion status of the network cannot be determined through the use of round trip time measurements. Instead, it is determined by comparing a flow's ingress and egress rates. In the slow start phase, a flow is considered to be experiencing congestion when its current ingress rate exceeds its reported egress rate by a factor of eight. The reason for the choice of the value eight is that we found a delay of three round trip times is typically required for a change in the ingress rate to be fully reflected in the egress rate of a backward feedback packet. During this time, the flow may double its ingress rate three times, increasing it by at most a factor of eight. Similarly, in the congestion avoidance phase, a flow is considered to be experiencing congestion whenever its current ingress rate exceeds its reported egress rate by three MRC increments. The reasoning in this case is similar to the reasoning used in the slow start case, except that a flow in the congestion avoidance phase may only increase its ingress rate by at most three MRC increments during three round trip times.

Clearly, the steps taken to determine congestion when an asynchronous BF packet arrives are more tolerant of transient congestion than the steps taken to determine congestion when a normal BF packet arrives. This is because asynchronous BF packets are only meant to be used as a stopgap measure to prevent serious congestion from developing during the interval between normal BF packet arrivals.

Simulation parameter	Value
Packet size	1000 bytes
Router queue size	100 packets
Maximum segment size (MSS)	1500 bytes
TCP implementation	Reno [12]
TCP window size	100 kbytes
MRC factor (MF)	10
AsynchInterval	10 msec
TSW window size	10 msec
End-system-to-edge propagation delay	100 μsec
End-system-to-edge link bandwidth	10 Mbps

Table 1. Default simulation parameters

III. Simulation Experiments

In this section, we present the results of several simulation experiments, each of which is designed to test a different aspect of Network Border Patrol. The first set of experiments examines the ability of NBP to prevent congestion collapse, and the second set of experiments examines its ability to provide fair bandwidth allocations to competing network flows. All simulations were run for 100 seconds using the UC Berkeley/LBNL/VINT ns-2 simulator [13]. The ns-2 code implementing NBP and the scripts to run these simulations are available at the UCI Network Research Group web site [14]. Default simulation parameters are shown in Table 1. They are set to values commonly used in the Internet and are used in all simulation experiments unless otherwise noted.

A. Preventing Congestion Collapse

The first set of simulation experiments explores NBP's ability to prevent congestion collapse from undelivered packets. In the first experiment, we study the scenario depicted in Figure 7. One

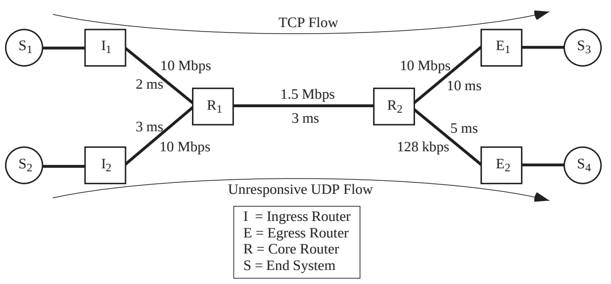


Fig. 7. A network with a single shared link

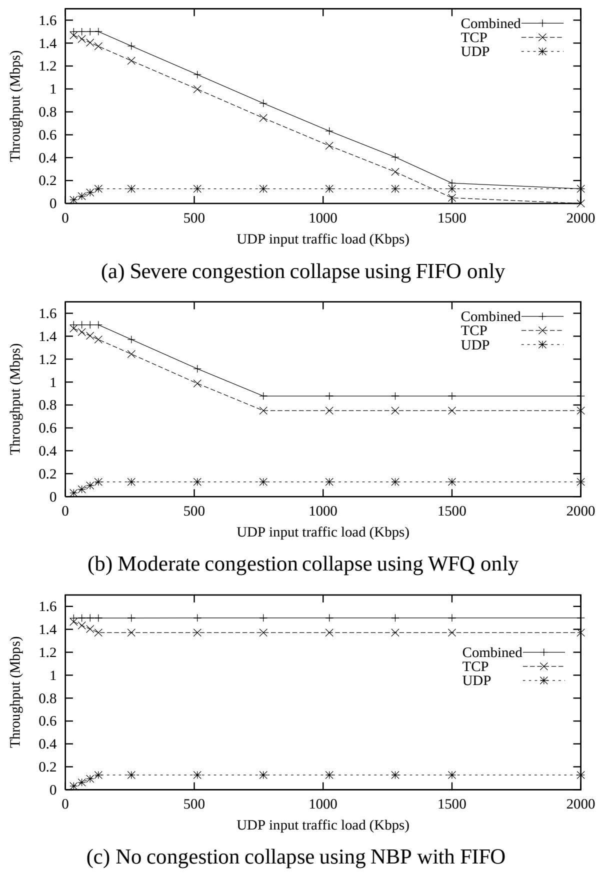


Fig. 8. Congestion collapse observed as unresponsive traffic load increases. The solid line shows the combined throughput delivered by the network.

flow is a TCP flow generated by an application which always has data to send, and the other flow is an unresponsive constant bit rate UDP flow. Both flows compete for access to a shared 1.5 Mbps bottleneck link (R_1-R_2), and only the UDP flow traverses a second bottleneck link (R_2-E_2), which has a limited capacity of 128 kbps.

Figure 8 shows the throughput achieved by the two flows as the UDP source's transmission rate is increased from 32 kbps to 2 Mbps. The combined throughput delivered by the network (i.e., the sum of both flow throughputs) is also shown. Three different cases are examined under this scenario. The first is the benchmark case used for comparison: NBP is not used between edge routers,

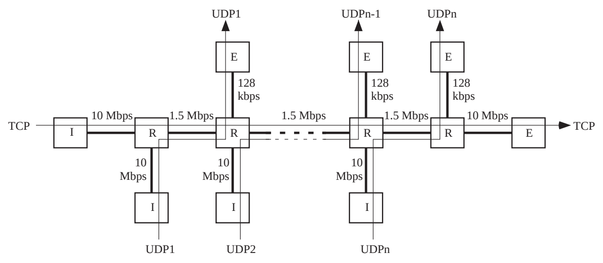


Fig. 9. A network with multiple congested router hops

and all routers schedule the delivery of packets on a FIFO basis. As Figure 8(a) shows, the network experiences severe congestion collapse as the UDP flow's transmission rate increases, because the UDP flow fails to respond adaptively to the discarding of its packets on the second bottleneck link. When the UDP load increases to 1.5 Mbps, the TCP flow's throughput drops nearly to zero. In the second case, weighted fair queueing replaces FIFO queueing in each of the routers, and the result, shown in Figure 8(b), is better throughput for the TCP flow. However, as indicated by the combined throughput of both flows, congestion collapse still occurs as the UDP load increases. Although WFQ allocates 750 kbps to both flows at the first bottleneck link, only 128 kbps of this bandwidth is successfully exploited by the UDP flow, which is even more seriously bottlenecked by a second link. The remaining 622 kbps is wasted on undelivered packets. In the third case, FIFO queues are reintroduced, and NBP is installed in the edge routers. As Figure 8(c) shows, NBP effectively eliminates congestion collapse; the TCP flow achieves a nearly optimal throughput of 1.37 Mbps, and the combined throughput remains very close to 1.5 Mbps.

In the second experiment, we examine whether these positive results continue to be demonstrated when a TCP flow traverses several bottleneck links carrying traffic from unresponsive UDP flows. The simulation model for this experiment is shown in Figure 9. In this configuration, a TCP flow shares several 1.5 Mbps bottleneck links with unresponsive UDP flows, each of which is further bottlenecked by another link with a capacity of 128 kbps. All links have propagation delays of 10 msec, and the UDP sources each transmit packets at a constant rate of 1 Mbps.

Figure 10 shows the throughput of the TCP flow as the number of congested router hops increases from 1 to 10. When only FIFO scheduling is used, the TCP flow achieves a throughput of approximately 0.5 Mbps regardless of the number of hops, whereas NBP allows the network to avoid congestion collapse, allocating nearly 1.31 Mbps to the TCP flow when the number of hops is small. As the number of hops increases, the throughput of the TCP flow diminishes somewhat due to increased feedback delays between the TCP flow's edge routers.

B. Achieving Fairness

The primary goal of NBP is to prevent congestion collapse from occurring. However, its secondary goal is to improve the fairness of bandwidth allocations to competing network flows. In this second set of simulation experiments, we examine whether NBP can

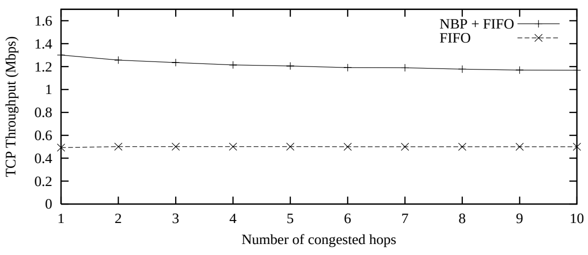
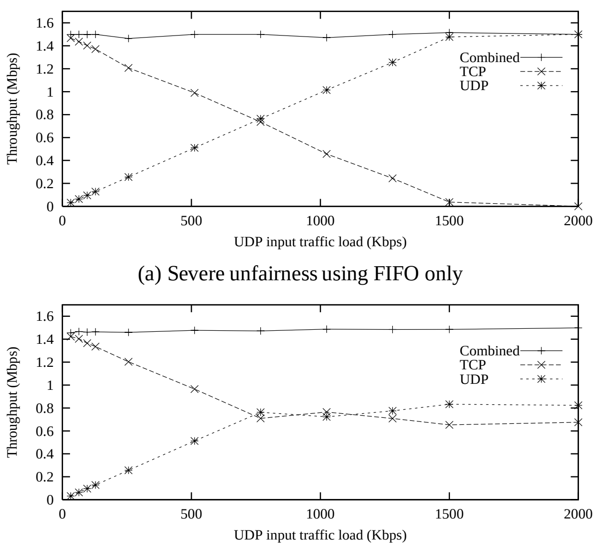


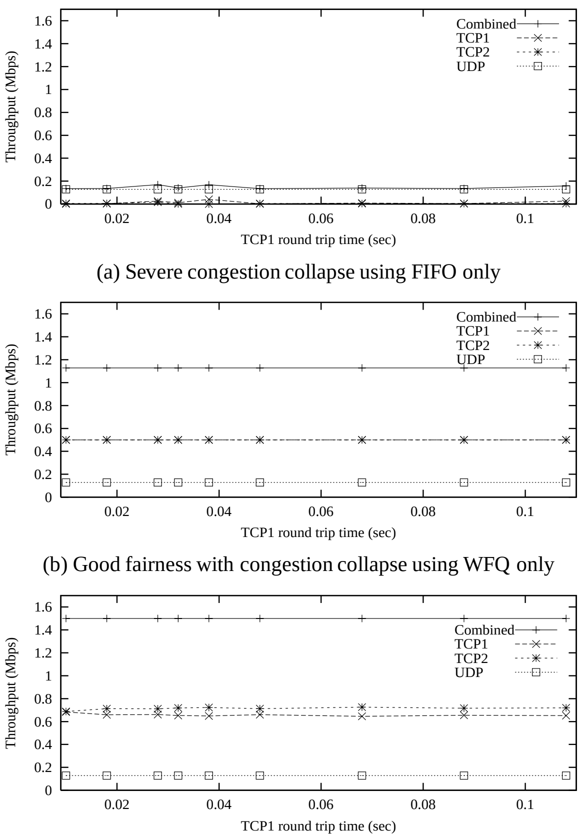
Fig. 10. TCP throughput in a network with multiple congested router hops



(a) Severe unfairness using FIFO only

(b) Moderate unfairness using NBP with FIFO

Fig. 11. Unfairness as the unresponsive traffic load increases



(a) Severe congestion collapse using FIFO only

(b) Good fairness with congestion collapse using WFQ only

(c) Slight unfairness but no congestion collapse using NBP with FIFO

Fig. 12. Unfairness as the TCP round trip time increases

achieve fair bandwidth allocations on its own, and, if not, whether it can do so in conjunction with other common network protocols and mechanisms.

In the first fairness experiment, we consider only one cause of unfairness: the existence of unresponsive flows. We return to the scenario depicted in Figure 7 but replace the second bottleneck link (R_2-E_2) with a higher capacity 10 Mbps link. The TCP flow is generated by an application which always has data to send, and the UDP flow is generated by an unresponsive source which transmits packets at a constant bit rate.

Since there is only one 1.5 Mbps bottleneck link (R_1-R_2) in this scenario, the max-min fair allocation of bandwidth between the flows is 750 kbps (if the UDP source exceeds a transmission rate of 750 kbps). However, as Figure 11(a) shows, fairness is clearly not achieved when only FIFO scheduling is used in routers. As the unresponsive UDP traffic load increases, the TCP flow experiences congestion and reduces its transmission rate, thereby granting an unfairly large amount of bandwidth to the unresponsive UDP flow. Thus, although there is no congestion collapse from undelivered packets, there is clearly unfairness. Figure 11(b) shows the throughput of each flow when NBP is introduced. Notice that NBP is able to reduce the amount of unfairness observed with FIFO scheduling only, but it does not completely eliminate unfairness. This is due to the fact that NBP has no mechanism that explicitly enforces fairness.

In the second fairness experiment we consider another cause of unfairness: TCP's dependence on the round trip time. In order to study this type of unfairness, we reuse the scenario from the first fairness experiment, but we return the second bottleneck link capacity to 128 kbps and introduce a new TCP flow (TCP2) between S_2 and S_3. Thus, two TCP flows and one unresponsive UDP flow share the first bottleneck link (R_1-R_2), and only the UDP flow crosses the second bottleneck link (R_2-E_2). In order to study the impact of increasing round trip times on fairness, the round trip time of the original TCP flow (TCP1) is varied by changing the propagation delay of link I_1-R_1. All other link propagation delays remain fixed as shown in Figure 7, and the transmission rate of the UDP source is set to 1.5 Mbps.

Figure 12(a) shows the resulting throughput of each flow when FIFO scheduling is used in all routers. Congestion collapse occurs to such an extent that both TCP flows achieve throughputs of zero, regardless of the round trip time of the TCP1 flow. Figure 12(b) depicts the throughput of each flow when FIFO scheduling is replaced with WFQ at all routers. WFQ allows the flows to achieve perfectly fair allocations of the bottleneck link bandwidth, but it does not prevent congestion collapse, as indicated by the fact that the combined throughput is less than 1.5 Mbps. Figure 12(c) shows the throughput of each flow when NBP is com-

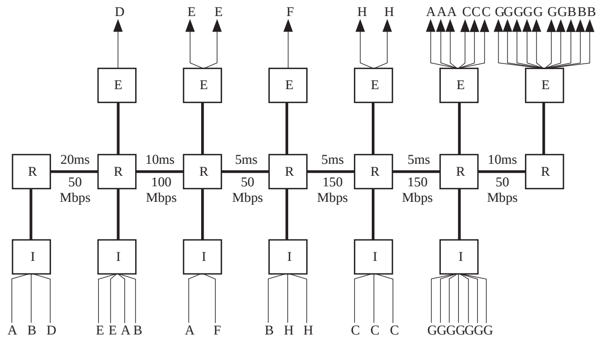


Fig. 13. The GFC-2 network

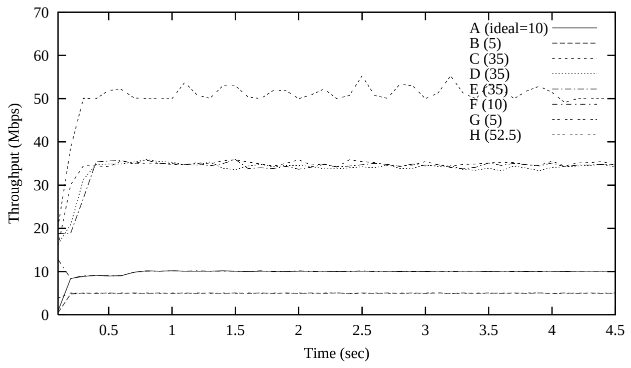
Flow Group	Ideal global max-min fair share	Simulation results: Throughput using WFQ only	Throughput using NBP with FIFO	Throughput using NBP with WFQ	Throughput using NBP with CSFQ
A	10	8.32	10.96	10.00	10.40
B	5	5.04	1.84	5.04	4.48
C	35	27.12	31.28	34.23	31.52
D	35	16.64	33.84	34.95	32.88
E	35	16.64	37.76	34.87	33.36
F	10	8.32	7.60	10.08	8.08
G	5	4.96	1.04	4.96	5.28
H	52.5	36.15	46.87	50.47	47.76

Table 2. Per-flow throughput in the GFC-2 network (in Mbps)

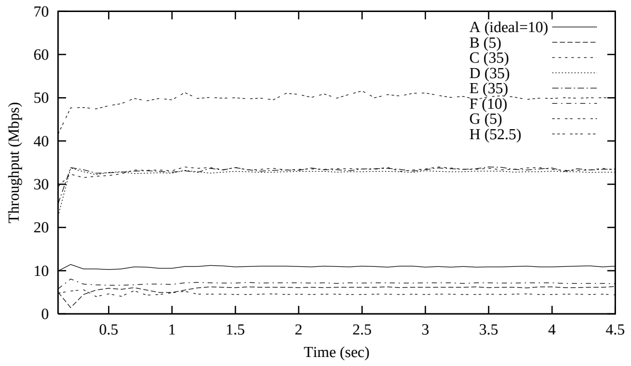
bined with FIFO scheduling. Although the combined throughput is very close to 1.5 Mbps and congestion collapse is prevented, NBP does not completely eliminate the unfair bandwidth allocations created by TCP1's longer round trip time.

In the third and final fairness experiment, we study whether NBP can be made more fair by combining it with a fair queueing mechanism such as weighted fair queueing or core-stateless fair queueing. We consider the network model shown in Figure 13. This model is adapted from the second General Fairness Configuration (GFC-2), which is specifically designed to test the max-min fairness of traffic control algorithms [15]. It consists of 22 unresponsive UDP flows, each generated by a source transmitting at a constant bit rate of 100 Mbps. Flows belong to flow groups which are labeled from A to H, and the network is designed in such a way that members of each flow group receive the same max-min bandwidth allocations. Links connecting core routers serve as bottlenecks for at least one of the 22 flows, and all links have propagation delays of 5 msec and bandwidths of 150 Mbps unless otherwise shown in the figure.

The first column of Table 2 lists the global max-min fair share allocations for all flows shown in Figure 13. These values represent the ideal bandwidth allocations for any traffic control mechanism that attempts to provide global max-min fairness. The remaining columns list the equilibrium-state throughputs actually observed after 4.5 seconds of simulation for several scenarios. (Only the results for a single member of each flow group are shown.) In the first scenario, NBP is not used and all routers perform WFQ. As indicated by comparing the values in the first and second columns, WFQ by itself is not able to achieve global max-min fairness for all flows. This is due to the fact that WFQ does not prevent congestion collapse. In the second scenario, NBP is introduced at edge routers and FIFO scheduling is assumed at all routers. Results listed in the third column show that NBP with



(a) Using NBP with WFQ



(b) Using NBP with CSFQ

Fig. 14. Per-flow throughput in the GFC-2 network

FIFO also fails to achieve global max-min fairness in the GFC-2 network, largely because NBP has no mechanism to explicitly enforce fairness. In the third and fourth simulation scenarios, NBP is combined with WFQ and CSFQ, respectively, and in both cases NBP is able to achieve bandwidth allocations that are approximately max-min fair for all flows.

NBP with WFQ achieves slightly better fairness than NBP with CSFQ. We suspect two reasons for this fact. First, CSFQ is an approximation of WFQ, and its performance depends on the accuracy of its estimation of a flow's input rate and fair share. Second, CSFQ's fairness mechanism engages only when congestion is detected (i.e., when a router's buffer occupancy becomes sufficiently large). Since NBP keeps buffer occupancies low by continuously monitoring and responding to variations in the edge-to-edge round trip time, CSFQ is not given many opportunities to engage.

Figures 14(a) and 14(b) show how rapidly the throughput of each flow converges to its max-min fair bandwidth allocation for the NBP with WFQ and the NBP with CSFQ cases, respectively. Even in a complex network like the one simulated here, all flows converge to an approximately max-min fair bandwidth allocation within one second.

IV. Implementation Issues

As we saw in the previous section, Network Border Patrol is a congestion avoidance mechanism that effectively prevents congestion collapse and provides approximate max-min fairness when used with a fair queueing mechanism. However, a number of important implementation issues must be addressed before NBP can be feasibly deployed in the Internet. Among these issues are the following:

1. *Scalable flow classification.* Perhaps the biggest impediment to NBP's scalability is its reliance upon flow classification at edge routers. In a network with a large number of flows, the overheads of maintaining per-flow state, communicating per-flow feedback, and performing per-flow rate control and rate monitoring may become inordinately expensive. Fortunately, it is possible to address this concern by classifying flows more coarsely at edge routers. Instead of classifying a flow using the packet's addresses and port numbers, the network's edge routers may aggregate many flows together by, for instance, classifying flows using only the packet's address fields. Alternatively, they might choose to classify flows even more coarsely using only the packet's destination network address. Coarse-grained flow aggregation has the effect of significantly reducing the number of flows seen by NBP edge routers. However, its drawback is that adaptive flows aggregated with unresponsive flows may be indiscriminately punished by an ingress router. Hence, NBP flow aggregation creates a trade-off between scalability and per-flow fairness.

2. *Scalable inter-domain deployment.* Another approach to improving the scalability of NBP, inspired by a suggestion in [7], is to develop trust relationships between domains that deploy NBP. The inter-domain router connecting two or more mutually trusting domains may then become a simple NBP core router with no need to perform per-flow tasks or keep per-flow state. If a trust relationship cannot be established, border routers between the two domains may exchange congestion information, so that congestion collapse can be prevented not only within a domain, but throughout multiple domains.

3. *Scalable fairness.* Although simulation results show that NBP is able to achieve the best approximation to max-min fairness when it is combined with WFQ, WFQ requires that core routers perform per-flow operations, making it less scalable than CSFQ. In networks where only a moderate number of simultaneous flows is possible (e.g., a campus network), NBP with WFQ may be preferable for its better fairness. However, NBP with CSFQ is preferable in networks with a large number of flows since approximate global max-min fairness is achieved in a more scalable core-stateless fashion.

4. *Incremental deployment.* It is crucial that NBP be implemented in all edge routers of an NBP-capable network. If one ingress router fails to police arriving traffic or one egress router fails to monitor departing traffic, NBP will not operate correctly and congestion collapse will be possible. Nevertheless, it is not necessary for all *networks* in the Internet to deploy NBP in order for it to be effective. Any network that deploys NBP will enjoy the benefits of eliminated congestion collapse within the network. Hence, it is possible to incrementally deploy NBP into the Internet on a network-by-network basis.

5. *Multicast.* Multicast routing makes it possible for copies of a flow's packets to leave the network through more than one egress router. When this occurs, an NBP ingress router must examine backward feedback packets returning from each of the multicast flow's egress routers. To determine whether the multicast flow is experiencing congestion, the ingress router should execute its rate control algorithm using backward feedback packets from the most congested ingress-to-egress path (i.e., the one with the lowest flow egress rate). This has the effect of limiting the ingress rate of a multicast flow according to the most congested link in the flow's multicast tree.

6. *Multi-path routing.* Multi-path routing makes it possible for packets from a single flow to leave the network through different egress routers. In order to support this possibility, an NBP ingress router may need to examine backward feedback packets from more than one egress router in order to determine the combined egress rate for a single flow. For a flow passing through more than one egress router, its combined egress rate is equal to the sum of the flow's egress rates reported in backward feedback packets from each egress router.

7. *Integrated or differentiated services.* NBP treats all flows identically, but integrated and differentiated services networks allow flows to receive different qualities of service. In such networks, NBP should be used to regulate best effort flows only. Flows using network services other than best effort are likely to be policed by separate traffic control mechanisms.

V. Conclusion

In this paper, we have presented a novel congestion avoidance mechanism for the Internet called Network Border Patrol. Unlike existing Internet congestion control approaches, which rely solely on end-to-end control, NBP is able to prevent congestion collapse from undelivered packets. It does this by ensuring at the border of the network that each flow's packets do not enter the network faster than they are able to leave it. NBP requires no modifications to core routers nor to end systems. Only edge routers are enhanced so that they can perform the requisite per-flow monitoring, per-flow rate control and feedback exchange operations.

Extensive simulation results provided in this paper show that NBP successfully prevents congestion collapse from undelivered packets. They also show that, while NBP is unable to eliminate unfairness on its own, it is able to achieve approximate global max-min fairness for competing network flows when combined with a fair queueing mechanism such as WFQ. Furthermore, NBP, when combined with CSFQ, approximates global max-min fairness in a completely core-stateless fashion.

As in any feedback-based traffic control mechanism, stability is an important performance concern in NBP. Using techniques described in [16], we plan as part of our future work to perform an analytical study of NBP's stability and convergence toward max-min fairness. Preliminary results already suggest that NBP benefits greatly from its use of explicit rate feedback, which prevents rate over-corrections in response to indications of network congestion.

Acknowledgments

We would like to thank Ion Stoica, Scott Shenker and Hui Zhang for the *ns*-2 core-stateless fair queueing code and also Paolo Losi for the *ns*-2 weighted fair queueing code.

REFERENCES

[1] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, August 1999, To appear.

[2] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *Proc. of ACM SIGCOMM*, September 1998, pp. 303–314.

[3] B. Suter, T.V. Lakshman, D. Stiliadis, and A. Choudhury, "Design Considerations for Supporting TCP with Per-Flow Queueing," in *Proc. of IEEE Infocom '98*, March 1998, pp. 299–305.

[4] B. Braden *et al.*, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, IETF, April 1998.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," in *Proc. of ACM SIGCOMM*, September 1989, pp. 1–12.

[6] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control – the Single Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

[7] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proc. of ACM SIGCOMM*, September 1998, pp. 118–130.

[8] D. Lin and R. Morris, "Dynamics of Random Early Detection," in *Proc. of ACM SIGCOMM*, September 1997, pp. 127–137.

[9] D. Bertsekas and R. Gallager, *Data Networks, second edition*, Prentice Hall, 1987.

[10] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, and R. Viswanathan, "ERICA Switch Algorithm: A Complete Description," ATM Forum Document 96-1172, Traffic Management WG, August 1996.

[11] D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[12] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," RFC 2001, IETF, January 1997.

[13] LBNL Network Research Group, *UCB/LBNL/VINT Network Simulator - ns (version 2)*, http://www-mash.cs.berkeley.edu/ns/, September 1997.

[14] UCI Network Research Group, *Network Border Patrol (NBP)*, http://netresearch.ics.uci.edu/nbp/, 1999.

[15] B. Vandalore, S. Fahmy, R. Jain, R. Goyal, and M. Goyal, "A Definition of Generalized Fairness and its Support in Switch Algorithms," ATM Forum Document 98-0151, Traffic Management WG, February 1998.

[16] W.K. Tsai and Y. Kim, "Re-Examining Maxmin Protocols: A Fundamental Study on Convergence, Complexity, Variations, and Performance," in *Proc. of IEEE Infocom*, April 1999, pp. 811–818.