Network Working Group
Request for Comments: 1254

A. Mankin
MITRE
K. Ramakrishnan
Digital Equipment Corporation
Editors

August 1991

Gateway Congestion Control Survey

Status of this Memo

This memo provides information for the Internet community. It is a survey of some of the major directions and issues. It does not specify an Internet standard. Distribution of this memo is unlimited.

Abstract

The growth of network intensive Internet applications has made gateway congestion control a high priority. The IETF Performance and Congestion Control Working Group surveyed and reviewed gateway congestion control and avoidance approaches. The purpose of this paper is to present our review of the congestion control approaches, as a way of encouraging new discussion and experimentation. Included in the survey are Source Quench, Random Drop, Congestion Indication (DEC Bit), and Fair Queueing. The task remains for Internet implementors to determine and agree on the most effective mechanisms for controlling gateway congestion.

1. Introduction

Internet users regularly encounter congestion, often in mild forms. However, severe congestion episodes have been reported also; and gateway congestion remains an obstacle for Internet applications such as scientific supercomputing data transfer. The need for Internet congestion control originally became apparent during several periods of 1986 and 1987, when the Internet experienced the "congestion collapse" condition predicted by Nagle [Nag84]. A large number of widely dispersed Internet sites experienced simultaneous slowdown or cessation of networking services for prolonged periods. BBN, the firm responsible for maintaining the then backbone of the Internet, the ARPANET, responded to the collapse by adding link capacity [Gar87].

Much of the Internet now uses as a transmission backbone the National Science Foundation Network (NSFNET). Extensive monitoring and capacity planning are being done for the NSFNET backbone; still, as

the demand for this capacity grows, and as resource-intensive applications such as wide-area file system management [Sp89] increasingly use the backbone, effective congestion control policies will be a critical requirement.

Only a few mechanisms currently exist in Internet hosts and gateways to avoid or control congestion. The mechanisms for handling congestion set forth in the specifications for the DoD Internet protocols are limited to:

Window flow control in TCP [Pos81b], intended primarily for controlling the demand on the receiver's capacity, both in terms of processing and buffers.

Source quench in ICMP, the message sent by IP to request that a sender throttle back [Pos81a].

One approach to enhancing Internet congestion control has been to overlay the simple existing mechanisms in TCP and ICMP with more powerful ones. Since 1987, the TCP congestion control policy, Slow-start, a collection of several algorithms developed by Van Jacobson and Mike Karels [Jac88], has been widely adopted. Successful Internet experiences with Slow-start led to the Host Requirements RFC [HREQ89] classifying the algorithms as mandatory for TCP. Slow-start modifies the user's demand when congestion reaches such a point that packets are dropped at the gateway. By the time such overflows occur, the gateway is congested. Jacobson writes that the Slow-start policy is intended to function best with a complementary gateway policy [Jac88].

1.1 Definitions

The characteristics of the Internet that we are interested in include that it is, in general, an arbitrary mesh-connected network. The internetwork protocol is connectionless. The number of users that place demands on the network is not limited by any explicit mechanism; no reservation of resources occurs and transport layer set-ups are not disallowed due to lack of resources. A path from a source to destination host may have multiple hops, through several gateways and links. Paths through the Internet may be heterogeneous (though homogeneous paths also exist and experience congestion). That is, links may be of different speeds. Also, the gateways and hosts may be of different speeds or may be providing only a part of their processing power to communication-related activity. The buffers for storing information flowing through Internet gateways are finite. The nature of the internet protocol is to drop packets when these buffers overflow.

Gateway congestion arises when the demand for one or more of the resources of the gateway exceeds the capacity of that resource. The resources include transmission links, processing, and space used for buffering. Operationally, uncongested gateways operate with little queueing on average, where the queue is the waiting line for a particular resource of the gateway. One commonly used quantitative definition [Kle79] for when a resource is congested is when the operating point is greater than the point at which resource power is maximum, where resource power is defined as the ratio of throughput to delay (See Section 2.2). At this operating point, the average queue size is close to one, including the packet in service. Note that this is a long-term average queue size. Several definitions exist for the timescale of averaging for congestion detection and control, such as dominant round-trip time and queue regeneration cycle (see Section 2.1).

Mechanisms for handling congestion may be divided into two categories, congestion recovery and congestion avoidance. Congestion recovery tries to restore an operating state, when demand has already exceeded capacity. Congestion avoidance is preventive in nature. It tries to keep the demand on the network at or near the point of maximum power, so that congestion never occurs. Without congestion recovery, the network may cease to operate entirely (zero throughput), whereas the Internet has been operating without congestion avoidance for a long time. Overall performance may improve with an effective congestion avoidance mechanism. Even if effective congestion avoidance was in place, congestion recovery schemes would still be required, to retain throughput in the face of sudden changes (increase of demand, loss of resources) that can lead to congestion.

In this paper, the term "user" refers to each individual transport (TCP or another transport protocol) entity. For example, a TCP connection is a "user" in this terminology. The terms "flow" and "stream" are used by some authors in the same sense. Some of the congestion control policies discussed in this paper, such as Selective Feedback Congestion Indication and Fair Queueing aggregate multiple TCP connections from a single host (or between a source host-destination host pair) as a virtual user.

The term "cooperating transport entities" will be defined as a set of TCP connections (for example) which follow an effective method of adjusting their demand on the Internet in response to congestion. The most restrictive interpretation of this term is that the transport entities follow identical algorithms for congestion control and avoidance. However, there may be some variation in these algorithms. The extent to which heterogeneous end-system congestion control and avoidance may be accommodated by gateway policies should

be a subject of future research. The role played in Internet performance of non-cooperating transport entities is discussed in Section 5.

1.2 Goals and Scope of This Paper

The task remains for Internet implementors to determine effective mechanisms for controlling gateway congestion. There has been minimal common practice on which to base recommendations for Internet gateway congestion control. In this survey, we describe the characteristics of one experimental gateway congestion management policy, Random Drop, and several that are better-known: Source Quench, Congestion Indication, Selective Feedback Congestion Indication, and Fair Queueing, both Bit-Round and Stochastic. A motivation for documenting Random Drop is that it has as primary goals low overhead and suitability for scaling up for Internets with higher speed links. Both of these are important goals for future gateway implementations that will have fast links, fast processors, and will have to serve large numbers of interconnected hosts.

The structure of this paper is as follows. First, we discuss performance goals, including timescale and fairness considerations. Second, we discuss the gateway congestion control policies. Random Drop is sketched out, with a recommendation for using it for congestion recovery and a separate section on its use as congestion avoidance. Third, since gateway congestion control in itself does not change the end-systems' demand, we briefly present the effective responses to these policies by two end-system congestion control schemes, Slow-start and End-System Congestion Indication. Among our conclusions, we address the issues of transport entities that do not cooperate with gateway congestion control. As an appendix, because of the potential interactions with gateway congestion policies, we report on a scheme to help in controlling the performance of Internet gateways to connection-oriented subnets (in particular, X.25).

Resources in the current Internet are not charged to users of them. Congestion avoidance techniques cannot be expected to help when users increase beyond the capacity of the underlying facilities. There are two possible solutions for this, increase the facilities and available bandwidth, or forcibly reduce the demand. When congestion is persistent despite implemented congestion control mechanisms, administrative responses are needed. These are naturally not within the scope of this paper. Also outside the scope of this paper are routing techniques that may be used to relocate demand away from congested individual resources (e.g., path-splitting and load-balancing).

2. Performance Goals

To be able to discuss design and use of various mechanisms for improving Internetwork performance, we need to have clear performance goals for the operation of gateways and sets of end-systems. Internet experience shows that congestion control should be based on adaptive principles; this requires efficient computation of metrics by algorithms for congestion control. The first issue is that of the interval over which these metrics are estimated and/or measured.

2.1 Interval for Measurement/Estimation of Performance Metrics

Network performance metrics may be distorted if they are computed over intervals that are too short or too long relative to the dynamic characteristics of the network. For instance, within a small interval, two FTP users with equal paths may appear to have sharply different demands, as an effect of brief, transient fluctuations in their respective processing. An overly long averaging interval results in distortions because of the changing number of users sharing the resource measured during the time. It is similarly important for congestion control mechanisms exerted at end systems to find an appropriate interval for control.

The first approach to the monitoring, or averaging, interval for congestion control is one based on round-trip times. The rationale for it is as follows: control mechanisms must adapt to changes in Internet congestion as quickly as possible. Even on an uncongested path, changed conditions will not be detected by the sender faster than a round-trip time. The effect of a sending end-system’s control will also not be seen in less than a round-trip time in the entire path as well as at the end systems. For the control mechanism to be adaptive, new information on the path is needed before making a modification to the control exerted. The statistics and metrics used in congestion control must be able to provide information to the control mechanism so that it can make an informed decision. Transient information which may be obsolete before a change is made by the end-system should be avoided. This implies the monitoring/estimating interval is one lasting one or more round trips. The requirements described here give bounds on:

How short an interval: not small enough that obsolete information is used for control;

How long: not more than the period at which the end-system makes changes.

But, from the point of view of the gateway congestion control policy, what is a round-trip time? If all the users of a given gateway have

the same path through the Internet, they also have the same round-trip time through the gateway. But this is rarely the case.

A meaningful interval must be found for users with both short and long paths. Two approaches have been suggested for estimating this dynamically, queue regeneration cycle and frequency analysis.

Use of the queue regeneration cycle has been described as part of the Congestion Indication policy. The time period used for averaging here begins when a resource goes from the idle to busy state. The basic interval for averaging is a "regeneration cycle" which is in the form of busy and idle intervals. Because an average based on a single previous regeneration may become old information, the recommendation in [JRC87] is to average over the sum of two intervals, that is, the previous (busy and idle) period, and the time since the beginning of the current busy period.

If the gateway users are window-based transport entities, it is possible to see how the regeneration interval responds to their round-trip times. If a user with a long round-trip time has the dominant traffic, the queue length may be zero only when that user is awaiting acknowledgements. Then the users with short paths will receive gateway congestion information that is averaged over several of their round-trip times. If the short path traffic dominates the activity in the gateway, i.e., the connections with shorter round-trip times are the dominant users of the gateway resources, then the regeneration interval is shorter and the information communicated to them can be more timely. In this case, users with longer paths receive, in one of their round-trip times, multiple samples of the dominant traffic; the end system averaging is based on individual user’s intervals, so that these multiple samples are integrated appropriately for these connections with longer paths.

The use of frequency analysis has been described by [Jac89]. In this approach, the gateway congestion control is done at intervals based on spectral analysis of the traffic arrivals. It is possible for users to have round-trip times close to each other, but be out of phase from each other. A spectral analysis algorithm detects this. Otherwise, if multiple round-trip times are significant, multiple intervals will be identified. Either one of these will be predominant, or several will be comparable. An as yet difficult problem for the design of algorithms accomplishing this technique is the likelihood of "locking" to the frequency of periodic traffic of low intensity, such as routing updates.

2.2 Power and its Relationship to the Operating Point

Performance goals for a congestion control/avoidance strategy embody a conflict in that they call for as high a throughput as possible, with as little delay as possible. A measure that is often used to reflect the tradeoff between these goals is power, the ratio of throughput to delay. We would like to maximize the value of power for a given resource. In the standard expression for power,

```
Power = (Throughput^alpha)/Delay
```

the exponent alpha is chosen for throughput, based on the relative emphasis placed on throughput versus delay: if throughput is more important, then a value of A alpha greater than one is chosen. If throughput and delay are equally important (e.g., both bulk transfer traffic and interactive traffic are equally important), then alpha equal to one is chosen. The operating point where power is maximized is the "knee" in the throughput and delay curves. It is desirable that the operating point of the resource be driven towards the knee, where power is maximized. A useful property of power is that it is decreasing whether the resource is under- or over-utilized relative to the knee.

In an internetwork comprising nodes and links of diverse speeds and utilization, bottlenecks or concentrations of demand may form. Any particular user can see a single bottleneck, which is the slowest or busiest link or gateway in the path (or possibly identical "balanced" bottlenecks). The throughput that the path can sustain is limited by the bottleneck. The delay for packets through a particular path is determined by the service times and queueing at each individual hop. The queueing delay is dominated by the queueing at the bottleneck resource(s). The contribution to the delay over other hops is primarily the service time, although the propagation delay over certain hops, such as a satellite link, can be significant. We would like to operate all shared resources at their knee and maximize the power of every user’s bottleneck.

The above goal underscores the significance of gateway congestion control. If techniques can be found to operate gateways at their resource knee, it can improve Internet performance broadly.

2.3 Fairness

We would like gateways to allocate resources fairly to users. A concept of fairness is only relevant when multiple users share a gateway and their total demand is greater than its capacity. If demands were equal, a fair allocation of the resource would be to provide an equal share to each user. But even over short intervals,

demands are not equal. Identifying the fair share of the resource for the user becomes hard. Having identified it, it is desirable to allocate at least this fair share to each user. However, not all users may take advantage of this allocation. The unused capacity can be given to other users. The resulting final allocation is termed a maximally fair allocation. [RJC87] gives a quantitative method for comparing the allocation by a given policy to the maximally fair allocation.

It is known that the Internet environment has heterogeneous transport entities, which do not follow the same congestion control policies (our definition of cooperating transports). Then, the controls given by a gateway may not affect all users and the congestion control policy may have unequal effects. Is "fairness" obtainable in such a heterogeneous community? In Fair Queueing, transport entities with differing congestion control policies can be insulated from each other and each given a set share of gateway bandwidth.

It is important to realize that since Internet gateways cannot refuse new users, fairness in gateway congestion control can lead to all users receiving small (sub-divided) amounts of the gateway resources inadequate to meet their performance requirements. None of the policies described in this paper currently addresses this. Then, there may be policy reasons for unequal allocation of the gateway resources. This has been addressed by Bit-Round Fair Queueing.

2.4 Network Management

Network performance goals may be assessed by measurements in either the end-system or gateway frame of reference. Performance goals are often resource-centered and the measurement of the global performance of "the network," is not only difficult to measure but is also difficult to define. Resource-centered metrics are more easily obtained, and do not require synchronization. That resource-centered metrics are appropriate ones for use in optimization of power is shown by [Jaf81].

It would be valuable for the goal of developing effective gateway congestion handling if Management Information Base (MIB) objects useful for evaluating gateway congestion were developed. The reflections on the control interval described above should be applied when network management applications are designed for this purpose. In particular, obtaining an instantaneous queue length from the managed gateway is not meaningful for the purposes of congestion management.

3. Gateway Congestion Control Policies

There have been proposed a handful of approaches to dealing with congestion in the gateway. Some of these are Source Quench, Random Drop, Congestion Indication, Selective Feedback Congestion Indication, Fair Queueing, and Bit-Round Fair Queueing. They differ in whether they use a control message, and indeed, whether they view control of the end-systems as necessary, but none of them in itself lowers the demand of users and consequent load on the network. End-system policies that reduce demand in conjunction with gateway congestion control are described in Section 4.

3.1 Source Quench

The method of gateway congestion control currently used in the Internet is the Source Quench message of the RFC-792 [Pos81a] Internet Control Message Protocol (ICMP). When a gateway responds to congestion by dropping datagrams, it may send an ICMP Source Quench message to the source of the dropped datagram. This is a congestion recovery policy.

The Gateway Requirements RFC, RFC-1009 [GREQ87], specifies that gateways should only send Source Quench messages with a limited frequency, to conserve CPU resources during the time of heavy load. We note that operating the gateway for long periods under such loaded conditions should be averted by a gateway congestion control policy. A revised Gateway Requirements RFC is being prepared by the IETF.

Another significant drawback of the Source Quench policy is that its details are discretionary, or, alternatively, that the policy is really a family of varied policies. Major Internet gateway manufacturers have implemented a variety of source quench frequencies. It is impossible for the end-system user on receiving a Source Quench to be certain of the circumstances in which it was issued. This makes the needed end-system response problematic: is the Source Quench an indication of heavy congestion, approaching congestion, a burst causing massive overload, or a burst slightly exceeding reasonable load?

To the extent that gateways drop the last arrived datagram on overload, Source Quench messages may be distributed unfairly. This is because the position at the end of the queue may be unfairly often occupied by the packets of low demand, intermittent users, since these do not send regular bursts of packets that can preempt most of the queue space.

[Fin89] developed algorithms for when to issue Source Quench and for responding to it with a rate-reduction in the IP layer on the sending

host. The system controls end-to-end performance of connections in a manner similar to the congestion avoidance portion of Slow-start TCP [Jac88].

3.2 Random Drop

Random Drop is a gateway congestion control policy intended to give feedback to users whose traffic congests the gateway by dropping packets on a statistical basis. The key to this policy is the hypothesis that a packet randomly selected from all incoming traffic will belong to a particular user with a probability proportional to the average rate of transmission of that user. Dropping a randomly selected packet results in users which generate much traffic having a greater number of packets dropped compared with those generating little traffic. The selection of packets to be dropped is completely uniform. Therefore, a user who generates traffic of an amount below the "fair share" (as defined in Section 2.3) may also experience a small amount of packet loss at a congested gateway. This character of uniformity is in fact a primary goal that Random Drop attempts to achieve.

The other primary goal that Random Drop attempts to achieve is a theoretical overhead which is scaled to the number of shared resources in the gateway rather than the number of its users. If a gateway congestion algorithm has more computation the more users there are, this can lead to processing demands that are higher as congestion increases. Also the low-overhead goal of Random Drop addresses concerns about the scale of gateway processing that will be required in the mid-term Internet as gateways with fast processors and links are shared by very large active sets of users.

3.2.1 For Congestion Recovery

Random Drop has been proposed as an improvement to packet dropping at the operating point where the gateway’s packet buffers overflow. This is using Random Drop strictly as a congestion recovery mechanism.

In Random Drop congestion recovery, instead of dropping the last packet to arrive at the queue, a packet is selected randomly from the queue. Measurements of an implementation of Random Drop Congestion Recovery [Man90] showed that a user with a low demand, due to a longer round-trip time path than other users of the gateway, had a higher drop rate with RDCR than without. The throughput accorded to users with the same round-trip time paths was nearly equal with RDCR as compared to without it. These results suggest that RDCR should be avoided unless it is used within a scheme that groups traffic more or less by round-trip time.

3.2.2 For Congestion Avoidance

Random Drop is also proposed as a congestion avoidance policy [Jac89]. The intent is to initiate dropping packets when the gateway is anticipated to become congested and remain so unless there is some control exercised. This implies selection of incoming packets to be randomly dropped at a rate derived from identifying the level of congestion at the gateway. The rate is the number of arrivals allowed between drops. It depends on the current operating point and the prediction of congestion.

A part of the policy is to determine that congestion will soon occur and that the gateway is beginning to operate beyond the knee of the power curve. With a suitably chosen interval (Section 2.1), the number of packets from each individual user in a sample over that interval is proportional to each user's demand on the gateway. Then, dropping one or more random packets indicates to some user(s) the need to reduce the level of demand that is driving the gateway beyond the desired operating point. This is the goal that a policy of Random Drop for congestion avoidance attempts to achieve.

There are several parameters to be determined for a Random Drop congestion avoidance policy. The first is an interval, in terms of number of packet arrivals, over which packets are dropped with uniform probability. For instance, in a sample implementation, if this interval spanned 2000 packet arrivals, and a suitable probability of drop was 0.001, then two random variables would be drawn in a uniform distribution in the range of 1 to 2,000. The values drawn would be used by counting to select the packets dropped at arrival. The second parameter is the value for the probability of drop. This parameter would be a function of an estimate of the number of users, their appropriate control intervals, and possibly the length of time that congestion has persisted. [Jac89] has suggested successively increasing the probability of drop when congestion persists over multiple control intervals. The motivation for increasing the packet drop probability is that the implicit estimate of the number of users and random selection of their packets to drop does not guarantee that all users have received enough signals to decrease demand. Increasing the probability of drop increases the probability that enough feedback is provided. Congestion detection is also needed in Random Drop congestion avoidance, and could be implemented in a variety of ways. The simplest is a static threshold, but dynamically averaged measures of demand or utilization are suggested.

The packets dropped in Random Drop congestion avoidance would not be from the initial inputs to the gateway. We suggest that they would be selected only from packets destined for the resource which is

predicted to be approaching congestion. For example, in the case of a gateway with multiple outbound links, access to each individual link is treated as a separate resource, the Random Drop policy is applied at each link independently. Random Drop congestion avoidance would provide uniform treatment of all cooperating transport users, even over individual patterns of traffic multiplexed within one user's stream. There is no aggregation of users.

Simulation studies [Zha89], [Has90] have presented evidence that Random Drop is not fair across cooperating and non-cooperating transport users. A transport user whose sending policies included Go-Back-N retransmissions and did not include Slow-start received an excessive share of bandwidth from a simple implementation of Random Drop. The simultaneously active Slow-start users received unfairly low shares. Considering this, it can be seen that when users do not respond to control, over a prolonged period, the Random Drop congestion avoidance mechanism would have an increased probability of penalizing users with lower demand. Their packets dropped, these users exert the controls leading to their giving up bandwidth.

Another problem can be seen to arise in Random Drop [She89] across users whose communication paths are of different lengths. If the path spans congested resources at multiple gateways, then the user's probability of receiving an unfair drop and subsequent control (if cooperating) is exponentially increased. This is a significant scaling problem.

Unequal paths cause problems for other congestion avoidance policies as well. Selective Feedback Congestion Indication was devised to enhance Congestion Indication specifically because of the problem of unequal paths. In Fair Queueing by source-destination pairs, each path gets its own queue in all the gateways.

3.3 Congestion Indication

The Congestion Indication policy is often referred to as the DEC Bit policy. It was developed at DEC [JRC87], originally for the Digital Network Architecture (DNA). It has also been specified for the congestion avoidance of the ISO protocols TP4 and CLNP [NIST88]. Like Source Quench, it uses explicit communications from the congested gateway to the user. However, to use the lowest possible network resources for indicating congestion, the information is communicated in a single bit, the Congestion Experienced Bit, set in the network header of the packets already being forwarded by a gateway. Based on the condition of this bit, the end-system user makes an adjustment to the sending window. In the NSP transport protocol of DECNET, the source makes an adjustment to its window; in the ISO transport protocol, TP4, the destination makes this

adjustment in the window offered to the sender.

This policy attempts to avoid congestion by setting the bit whenever the average queue length over the previous queue regeneration cycle plus part of the current cycle is one or more. The feedback is determined independently at each resource.

3.4 Selective Feedback Congestion Indication

The simple Congestion Indication policy works based upon the total demand on the gateway. The total number of users or the fact that only a few of the users might be causing congestion is not considered. For fairness, only those users who are sending more than their fair share should be asked to reduce their load, while others could attempt to increase where possible. In Selective Feedback Congestion Indication, the Congestion Experienced Bit is used to carry out this goal.

Selective Feedback works by keeping a count of the number of packets sent by different users since the beginning of the queue averaging interval. This is equivalent to monitoring their throughputs. Based on the total throughput, a fair share for each user is determined and the congestion bit is set, when congestion approaches, for the users whose demand is higher than their fair share. If the gateway is operating below the throughput-delay knee, congestion indications are not set.

A min-max algorithm used to determine the fair share of capacity and other details of this policy are described in [RJC87]. One parameter to be computed is the capacity of each resource to be divided among the users. This metric depends on the distribution of inter-arrival times and packet sizes of the users. Attempting to determine these in real time in the gateway is unacceptable. The capacity is instead estimated from on the throughput seen when the gateway is operating in congestion, as indicated by the average queue length. In congestion (above the knee), the service rate of the gateway limits its throughput. Multiplying the throughput obtained at this operating point by a capacity factor (between 0.5 and 0.9) to adjust for the distributions yields an acceptable capacity estimate in simulations.

Selective Feedback Congestion Indication takes as input a vector of the number of packets sent by each source-destination pair of end-systems. Other alternatives include 1) destination address, 2) input/output link, and 3) transport connection (source/destination addresses and ports).

These alternatives give different granularities for fairness. In the

case where paths are the same or round-trip times of users are close together, throughputs are equalized similarly by basing the selective feedback on source or destination address. In fact, if the RTTs are close enough, the simple congestion indication policy would result in a fair allocation. Counts based on source/destination pairs ensure that paths with different lengths and network conditions get a fair throughput at the individual gateways. Counting packets based on link pairs has a low overhead, but may result in unfairness to users whose demand is below the fair share and are using a long path. Counts based on transport layer connection identifiers, if this information was available to Internet gateways, would make good distinctions, since the differences of demand of different applications and instances of applications would be separately monitored.

Problems with Selective Feedback Congestion Indication include that the gateway has significant processing to do. With the feasible choice of aggregation at the gateway, unfairness across users within the group is likely. For example, an interactive connection aggregated with one or more bulk transfer connections will receive congestion indications though its own use of the gateway resources is very low.

3.5 Fair Queueing

Fair Queueing is the policy of maintaining separate gateway output queues for individual end-systems by source-destination pair. In the policy as proposed by [Nag85], the gateway’s processing and link resources are distributed to the end-systems on a round-robin basis. On congestion, packets are dropped from the longest queue. This policy leads to equal allocations of resources to each source-destination pair. A source-destination pair that demands more than a fair share simply increases its own queueing delay and congestion drops.

3.5.1 Bit-Round Fair Queueing

An enhancement of Nagle Fair Queueing, the Bit-Round Fair Queuing algorithm described and simulated by [DKS89] addresses several shortcomings of Nagle’s scheme. It computes the order of service to packets using their lengths, with a technique that emulates a bit-by-bit round-robin discipline, so that long packets do not get an advantage over short ones. Otherwise the round-robin would be unfair, for example, giving more bandwidth to hosts whose traffic is mainly long packets than to hosts sourcing short packets.

The aggregation of users of a source-destination pair by Fair Queueing has the property of grouping the users whose round-trips are

similar. This may be one reason that the combination of Bit-Round Fair Queueing with Congestion Indication had particularly good simulated performance in [DKS89].

The aggregation of users has the expected drawbacks, as well. A TELNET user in a queue with an FTP user does not get delay benefits; and host pairs with high volume of connections get treated the same as a host pair with small number of connections and as a result gets unfair services.

A problem can be mentioned about Fair Queueing, though it is related to implementation, and perhaps not properly part of a policy discussion. This is a concern that the resources (processing) used for determining where to queue incoming packets would themselves be subject to congestion, but not to the benefits of the Fair Queuing discipline. In a situation where the gateway processor was not adequate to the demands on it, the gateway would need an alternative policy for congestion control of the queue awaiting processing. Clever implementation can probably find an efficient way to route packets to the queues they belong in before other input processing is done, so that processing resources can be controlled, too. There is in addition, the concern that bit-by-bit round FQ requires non-FCFS queueing even within the same source destination pairs to allow for fairness to different connections between these end systems.

Another potential concern about Fair Queueing is whether it can scale up to gateways with very large source-destination populations. For example, the state in a Fair Queueing implementation scales with the number of active end-to-end paths, which will be high in backbone gateways.

3.5.2 Stochastic Fairness Queuing

Stochastic Fairness Queueing (SFQ) has been suggested as a technique [McK90] to address the implementation issues relating to Fair Queueing. The first overhead that is reduced is that of looking up the source-destination address pair in an incoming packet and determining which queue that packet will have to be placed in. SFQ does not require as many memory accesses as Fair Queueing to place the packet in the appropriate queue. SFQ is thus claimed to be more amenable to implementation for high-speed networks [McK90].

SFQ uses a simple hash function to map from the source-destination address pair to a fixed set of queues. Since the assignment of an address pair to a queue is probabilistic, there is the likelihood of multiple address pairs colliding and mapping to the same queue. This would potentially degrade the additional fairness that is gained with Fairness Queueing. If two or more address pairs collide, they would

continue to do so. To deal with the situation when such a collision occurs, SFQ periodically perturbs the hash function so that these address pairs will be unlikely to collide subsequently.

The price paid for achieving this implementation efficiency is that SFQ requires a potentially large number of queues (we must note however, that these queues may be organized orthogonally from the buffers in which packets are stored. The buffers themselves may be drawn from a common pool, and buffers in each queue organized as a linked list pointed to from each queue header). For example, [McK90] indicates that to get good, consistent performance, we may need to have up to 5 to 10 times the number of active source-destination pairs. In a typical gateway, this may require around 1000 to 2000 queues.

[McK90] reports simulation results with SFQ. The particular hash function that is studied is using the HDLC's cyclic redundancy check (CRC). The hash function is perturbed by multiplying each byte by a sequence number in the range 1 to 255 before applying the CRC. The metric considered is the standard deviation of the number of packets output per source-destination pair. A perfectly fair policy would have a standard deviation of zero and an unfair policy would have a large standard deviation. In the example studied (which has up to 20 source-destination (s-d) pairs going through a single overloaded gateway), SFQ with 1280 queues (i.e., 64 times the number of source-destination pairs), approaches about 3 times the standard deviation of Fairness Queueing. This must be compared to a FCFS queueing discipline having a standard deviation which is almost 175 times the standard deviation of Fairness Queueing.

It is conjectured in [McK90] that a good value for the interval in between perturbations of the hash function appears to be in the area between twice the queue flush time of the stochastic fairness queue and one-tenth the average conversation time between a source-destination pair.

SFQ also may alleviate the anticipated scaling problems that may be an issue with Fair Queueing by not strictly requiring the number of queues equal to the possible source-destination pairs that may be presenting a load on a particular gateway. However, SFQ achieves this property by trading off some of the fairness for implementation efficiency.

[McK90] examines alternative strategies for implementation of Fair Queueing and SFQ and estimates their complexity on common hardware platforms (e.g., a Motorola 68020). It is suggested that mapping an IP address pair may require around 24 instructions per packet for Fair Queueing in the best case; in contrast SFQ requires 10

instructions in the worst case. The primary source of this gain is that SFQ avoids a comparison of the s-d address pair on the packet to the identity of the queue header. The relative benefit of SFQ over Fair Queueing is anticipated to be greater when the addresses are longer.

SFQ offers promising implemenatation benefits. However, the price to be paid in terms of having a significantly larger number of queues (and the consequent data structures and their management) than the number of s-d pairs placing a load on the gateway is a concern. SFQ is also attractive in that it may be used in concert with the DEC-bit scheme for Selective Feedback Congestion Indication to provide fairness as well as congestion avoidance.

4. END-SYSTEM CONGESTION CONTROL POLICIES

Ideally in gateway congestion control, the end-system transport entities adjust (decrease) their demand in response to control exerted by the gateway. Schemes have been put in practice for transport entities to adjust their demand dynamically in response to congestion feedback. To accomplish this, in general, they call for the user to gradually increase demand until information is received that the load on the gateway is too high. In response to this information, the user decreases load, then begins an exploratory increases again. This cycle is repeated continuously, with the goal that the total demand will oscillate around the optimal level.

We have already noted that a Slow-start connection may give up considerable bandwidth when sharing a gateway with aggressive transport entities. There is currently no way to enforce that end-systems use a congestion avoidance policy, though the Host Requirements RFC [HR89] has defined Slow-start as mandatory for TCP. This adverse effect on Slow-start connections is mitigated by the Fair Queueing policy. Our conclusions discuss further the coexistence of different end-system strategies.

This section briefly presents two fielded transport congestion control and avoidance schemes, Slow-start and End-System Congestion Indication, and the responses by means of which they cooperate with gateway policies. They both use the control paradigm described above. Slow-start, as mentioned in Section 1, was developed by [Jac88], and widely fielded in the BSD TCP implementation. End-system Congestion Indication was developed by [JRC87]. It is fielded in DEC’s Digital Network Architecture, and has been specified as well for ISO TP4 [NIST88].

Both Slow-start and End-system Congestion Indication view the relationship between users and gateways as a control system. They

have feedback and control components, the latter further broken down into a procedure bringing a new connection to equilibrium, and a procedure to maintain demand at the proper level. They make use of policies for increasing and decreasing the transport sender's window size. These require the sender to follow a set of self-restraining rules which dynamically adjust the send window whenever congestion is sensed.

A predecessor of these, CUTE, developed by [Jai86], introduced the use of retransmission timeouts as congestion feedback. The Slow-start scheme was also designed to use timeouts in the same way. The End-System policies for Congestion Indication use the Congestion Experienced Bit delivered in the network header as the primary feedback, but retransmission timeouts also provoke an end-system congestion response.

In reliable transport protocols like TCP and TP4, the retransmission timer must do its best to satisfy two conflicting goals. On one hand, the timer must rapidly detect lost packets. And, on the other hand, the timer must minimize false alarms. Since the retransmit timer is used as a congestion signal in these end-system policies, it is all the more important that timeouts reliably correspond to packet drops. One important rule for retransmission is to avoid bad sampling in the measurements that will be used in estimating the round-trip delay. [KP87] describes techniques to ensure accurate sampling. The Host Requirements RFC [HR89] makes these techniques mandatory for TCP.

The utilization of a resource can be defined as the ratio of its average arrival rate to its mean service rate. This metric varies between 0 and 1.0. In a state of congestion, one or more resources (link, gateway buffer, gateway CPU) has a utilization approaching 1.0. The average delay (round trip time) and its variance approach infinity. Therefore, as the utilization of a network increases, it becomes increasingly important to take into account the variance of the round trip time in estimating it for the retransmission timeout.

The TCP retransmission timer is based on an estimate of the round trip time. [Jac88] calls the round trip time estimator the single most important feature of any protocol implementation that expects to survive a heavy load. The retransmit timeout procedure in RFC-793 [Pos81b] includes a fixed parameter, beta, to account for the variance in the delay. An estimate of round trip time using the suggested values for beta is valid for a network which is at most 30% utilized. Thus, the RFC-793 retransmission timeout estimator will fail under heavy congestion, causing unnecessary retransmissions that add to the load, and making congestive loss detection impossible.

Slow-start TCP uses the mean deviation as an estimate of the variance

to improve the estimate. As a rough view of what happens with the Slow-start retransmission calculation, delays can change by approximately two standard deviations without the timer going off in a false alarm. The same method of estimation may be applicable to TP4. The procedure is:

```
Error     = Measured - Estimated
Estimated = Estimated + Gain_1 * Error
Deviation = Deviation + Gain_2 * (|Error| - Deviation)
Timeout   = Estimated + 2 * Deviation

Where:  Gain_1, Gain_2 are gain factors.
```

4.1 Response to No Policy in Gateway

Since packets must be dropped during congestion because of the finite buffer space, feedback of congestion to the users exists even when there is no gateway congestion policy. Dropping the packets is an attempt to recover from congestion, though it needs to be noted that congestion collapse is not prevented by packet drops if end-systems accelerate their sending rate in these conditions. The accurate detection of congestive loss by all retransmission timers in the end-systems is extremely important for gateway congestion recovery.

4.2 Response to Source Quench

Given that a Source Quench message has ambiguous meaning, but usually is issued for congestion recovery, the response of Slow-start to a Source Quench is to return to the beginning of the cycle of increase. This is an early response, since the Source Quench on overflow will also lead to a retransmission timeout later.

4.3 Response to Random Drop

The end-system’s view of Random Drop is the same as its view of loss caused by overflow at the gateway. This is a drawback of the use of packet drops as congestion feedback for congestion avoidance: the decrease policy on congestion feedback cannot be made more drastic for overflows than for the drops the gateway does for congestion avoidance. Slow-start responds rapidly to gateway feedback. In a case where the users are cooperating (all Slow-start), a desired outcome would be that this responsiveness would lead quickly to a decreased probability of drop. There would be, as an ideal, a self-adjusting overall amount of control.

4.4 Response to Congestion Indication

Since the Congestion Indication mechanism attempts to keep gateways uncongested, cooperating end-system congestion control policies need not reduce demand as much as with other gateway policies. The difference between the Slow-start response to packet drops and the End-System Congestion Indication response to Congestion Experienced Bits is primarily in the decrease policy. Slow-start decreases the window to one packet on a retransmission timeout. End-System Congestion Indication decreases the window by a fraction (for instance, to 7/8 of the original value), when the Congestion Experienced Bit is set in half of the packets in a sample spanning two round-trip times (two windows full).

4.5 Response to Fair Queuing

A Fair Queueing policy may issue control indications, as in the simulated Bit-Round Fair Queueing with DEC Bit, or it may depend only on the passive effects of the queueing. When the passive control is the main effect (perhaps because most users are not responsive to control indications), the behavior of retransmission timers will be very important. If retransmitting users send more packets in response to increases in their delay and drops, Fair Queueing will be prone to degraded performance, though collapse (zero throughput for all users) may be prevented for a longer period of time.

5. Conclusions

The impact of users with excessive demand is a driving force as proposed gateway policies come closer to implementation. Random Drop and Congestion Indication can be fair only if the transport entities sharing the gateway are all cooperative and reduce demand as needed. Within some portions of the Internet, good behavior of end-systems eventually may be enforced through administration. But for various reasons, we can expect non-cooperating transports to be a persistent population in the Internet. Congestion avoidance mechanisms will not be deployed all at once, even if they are adopted as standards. Without enforcement, or even with penalties for excessive demand, some end-systems will never implement congestion avoidance mechanisms.

Since it is outside the context of any of the gateway policies, we will mention here a suggestion for a relatively small-scale response to users which implement especially aggressive policies. This has been made experimentally by [Jac89]. It would implement a low priority queue, to which the majority of traffic is not routed. The candidate traffic to be queued there would be identified by a cache of recent recipients of whatever control indications the gateway

policy makes. Remaining in the cache over multiple control intervals is the criterion for being routed to the low priority queue. In approaching or established congestion, the bandwidth given to the low-service queue is decreased.

The goal of end-system cooperation itself has been questioned. As [She89] points out, it is difficult to define. A TCP implementation that retransmits before it determines that has been loss indicated and in a Go-Back-N manner is clearly non-cooperating. On the other hand, a transport entity with selective acknowledgement may demand more from the gateways than TCP, even while responding to congestion in a cooperative way.

Fair Queueing maintains its control of allocations regardless of the end-system congestion avoidance policies. [Nag85] and [DKS89] argue that the extra delays and congestion drops that result from misbehavior could work to enforce good end-system policies. Are the rewards and penalties less sharply defined when one or more misbehaving systems cause the whole gateway's performance to be less? While the tax on all users imposed by the "over-users" is much less than in gateways with other policies, how can it be made sufficiently low?

In the sections on Selective Feedback Congestion Indication and Bit-Round Fair Queueing we have pointed out that more needs to be done on two particular fronts:

How can increased computational overhead be avoided?

The allocation of resources to source-destination pairs is, in many scenarios, unfair to individual users. Bit-Round Fair Queueing offers a potential administrative remedy, but even if it is applied, how should the unequal allocations be propagated through multiple gateways?

The first question has been taken up by [McK90].

Since Selective Feedback Congestion Indication (or Congestion Indication used with Fair Queueing) uses a network bit, its use in the Internet requires protocol support; the transition of some portions of the Internet to OSI protocols may make such a change attractive in the future. The interactions between heterogeneous congestion control policies in the Internet will need to be explored.

The goals of Random Drop Congestion Avoidance are presented in this survey, but without any claim that the problems of this policy can be solved. These goals themselves, of uniform, dynamic treatment of users (streams/flows), of low overhead, and of good scaling

characteristics in large and loaded networks, are significant.

Appendix: Congestion and Connection-oriented Subnets

This section presents a recommendation for gateway implementation in an areas that unavoidably interacts with gateway congestion control, the impact of connection-oriented subnets, such as those based on X.25.

The need to manage a connection oriented service (X.25) in order to transport datagram traffic (IP) can cause problems for gateway congestion control. Being a pure datagram protocol, IP provides no information delimiting when a pair of IP entities need to establish a session between themselves. The solution involves compromise among delay, cost, and resources. Delay is introduced by call establishment when a new X.25 SVC (Switched Virtual Circuit) needs to be established, and also by queueing delays for the physical line. Cost includes any charges by the X.25 network service provider. Besides the resources most gateways have (CPU, memory, links), a maximum supported or permitted number of virtual circuits may be in contest.

SVCs are established on demand when an IP packet needs to be sent and there is no SVC established or pending establishment to the destination IP entity. Optionally, when cost considerations, and sufficient numbers of unused virtual circuits allow, redundant SVCs may be established between the same pair of IP entities. Redundant SVCs can have the effect of improving performance when coping with large end-to-end delay, small maximum packet sizes and small maximum packet windows. It is generally preferred though, to negotiate large packet sizes and packet windows on a single SVC. Redundant SVCs must especially be discouraged when virtual circuit resources are small compared with the number of destination IP entities among the active users of the gateway.

SVCs are closed after some period of inactivity indicates that communication may have been suspended between the IP entities. This timeout is significant in the operation of the interface. Setting the value too low can result in closing of the SVC even though the traffic has not stopped. This results in potentially large delays for the packets which reopen the SVC and may also incur charges associated with SVC calls. Also, clearing of SVCs is, by definition, nongraceful. If an SVC is closed before the last packets are acknowledged, there is no guarantee of delivery. Packet losses are introduced by this destructive close independent of gateway traffic and congestion.

When a new circuit is needed and all available circuits are currently

in use, there is a temptation to pick one to close (possibly using some Least Recently Used criterion). But if connectivity demands are larger than available circuit resources, this strategy can lead to a type of thrashing where circuits are constantly being closed and reopened. In the worst case, a circuit is opened, a single packet sent and the circuit closed (without a guarantee of packet delivery). To counteract this, each circuit should be allowed to remain open a minimum amount of time.

One possible SVC strategy is to dynamically change the time a circuit will be allowed to remain open based on the number of circuits in use. Three administratively controlled variables are used, a minimum time, a maximum time and an adaptation factor in seconds per available circuit. In this scheme, a circuit is closed after it has been idle for a time period equal to the minimum plus the adaptation factor times the number of available circuits, limited by the maximum time. By administratively adjusting these variables, one has considerable flexibility in adjusting the SVC utilization to meet the needs of a specific gateway.

Acknowledgements

This paper is the outcome of discussions in the Performance and Congestion Control Working Group between April 1988 and July 1989. Both PCC WG and plenary IETF members gave us helpful reviews of earlier drafts. Several of the ideas described here were contributed by the members of the PCC WG. The Appendix was written by Art Berggreen. We are particularly thankful also to Van Jacobson, Scott Shenker, Bruce Schofield, Paul McKenney, Matt Mathis, Geof Stone, and Lixia Zhang for participation and reviews.

References

[DKS89] Demers, A., Keshav, S., and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", Proceedings of SIGCOMM ’89.

[Fin89] Finn, G., "A Connectionless Congestion Control Algorithm", Computer Communications Review, Vol. 19, No. 5, October 1989.

[Gar87] Gardner, M., "BBN Report on the ARPANET", Proceedings of the McLean IETF, SRI-NIC IETF-87/3P, July 1987.

[GREQ87] Braden R., and J. Postel, "Requirements for Internet Gateways", RFC 1009, USC/Information Sciences Institute, June 1987.

[HREQ89] Braden R., Editor, "Requirements for Internet Hosts -- Communications Layers", RFC 1122, Internet Engineering Task Force, October 1989.

[Has90] Hashem, E., "Random Drop Congestion Control", M.S. Thesis, Massachusetts Institute of Technology, Department of Computer Science, 1990.

[Jac88] Jacobson, V., "Congestion Avoidance and Control", Proceedings of SIGCOMM ’88.

[Jac89] Jacobson, V., "Presentations to the IETF Performance and Congestion Control Working Group".

[Jaf81] Jaffe, J., "Bottleneck Flow Control", IEEE Transactions on Communications, COM-29(7), July, 1981.

[Jai86] Jain, R., "A Timeout-based Congestion Control Scheme for Window Flow-controlled Networks", IEEE Journal on Selected Areas in Communications, SAC-4(7), October 1986.

[JRC87] Jain, R., Ramakrishnan, K., and D. Chiu, "Congestion Avoidance in Computer Networks With a Connectionless Network Layer", Technical Report DEC-TR-506, Digital Equipment Corporation.

[Kle79] Kleinrock, L., "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications", Proceedings of the ICC ’79.

[KP87] Karn, P., and C. Partridge, "Improving Round Trip Estimates in Reliable Transport Protocols", Proceedings of SIGCOMM ’87.

[Man90] Mankin, A., "Random Drop Congestion Control", Proceedings of SIGCOMM ’90.

[McK90] McKenney, P., "Stochastic Fairness Queueing", Proceedings of INFOCOM ’90.

[Nag84] Nagle, J., "Congestion Control in IP/TCP Internetworks", RFC 896, FACC Palo Alto, 6 January 1984.

[Nag85] Nagle, J., "On Packet Switches With Infinite Storage", RFC 970, FACC Palo Alto, December 1985.

[NIST88] NIST, "Stable Implementation Agreements for OSI Protocols, Version 2, Edition 1", National Institute of Standards and Technology Special Publication 500-162, December 1988.

[Pos81a] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification", RFC-792, USC/Information Sciences Institute, September 1981.

[Pos81b] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC-793, DARPA, September 1981.

[RJC87] Ramakrishnan, K., Jain, R., and D. Chiu, "A Selective Binary Feedback Scheme for General Topologies", Technical Report DEC-TR-510, Digital Equipment Corporation.

[She89] Shenker, S., "Correspondence with the IETF Performance and Congestion Control Working Group".

[Sp89] Spector, A., and M. Kazar, "Uniting File Systems", Unix Review, Vol. 7, No. 3, March 1989.

[Zha89] Zhang, L., "A New Architecture for Packet Switching Network Protocols", Ph.D Thesis, Massachusetts Institute of Technology, Department of Computer Science, 1989.

Security Considerations

Security issues are not discussed in this memo.

Authors' Addresses

Allison Mankin
The MITRE Corporation
M/S W425
7525 Colshire Drive
McLean, VA 22102

Email: mankin@gateway.mitre.org

K.K. Ramakrishnan
Digital Equipment Corporation
M/S LKG1-2/A19
550 King Street
Littleton, MA 01754

Email: rama@kalvi.enet.dec.com