The Evolution of Congestion Control in TCP/IP: from Reactive Windows to Preventive Rate Control

Jim Martin
eNetwork Studies
Networking Systems
IBM Corporation

Arne Nilsson
Center for Advanced Computing and Communication
ECE Department
North Carolina State University

ABSTRACT

In a packet switched network, multiple sessions or users compete for network resources. Network resources include bandwidth (i.e., the transmission capacity of a link in the network), a router's switching capacity and a router's buffer capacity. If demand for a resource exceeds the capacity, congestion occurs. Symptoms of network congestion include increased packet delays or packet loss (from buffer overflow). Congestion control is the aspect of a networking protocol that defines how the network deals with congestion.

The congestion control schemes employed by the TCP/IP protocol have been widely studied. The mechanisms continue to be enhanced as TCP/IP evolves to meet new and more demanding requirements. It has been debated over the past decade as to whether window or rate control is more effective at controlling congestion. Recent work suggests that adding burst control to TCP provides relatively minor performance improvements. Furthermore it has been shown (through simulation) that the current TCP protocol is not able to efficiently utilize reserved bandwidth over an RSVP controlled-load connection. However, a set of enhancements have been proposed, the most significant being rate control, that address the limitations of TCP in an Integrated Services environment. Taking this one step further, we suggest that the performance of a rate-based version of TCP in an Integrated Services environment can be further improved with an additional end-to-end preventive congestion control scheme.

This paper is organized as follows. First, we provide a detailed discussion of congestion control in packet switched networks. Essentially we provide a framework that is used to describe and differentiate specific congestion control schemes throughout the paper. Next we cover TCP/IP congestion control, surveying the relevant research. Lastly we overview the Integrated Services Architecture focusing on congestion control and TCP performance issues.

1. Background on Congestion Control

1.1 Defining Congestion Control

In a packet switched network, multiple sessions or users compete for network resources. Network resources include bandwidth (i.e., the transmission capacity of a hop in the network), a router's switching, and a router's buffer capacity. If demand for a resource exceeds the capacity, congestion occurs. Symptoms of network congestion include increased packet delays or packet loss (from buffer overflow).

Congestion is not a static resource shortage problem but rather a dynamic allocation problem. For example, if the critical resource in a network is buffers in a router, adding more memory might not eliminate congestion. In fact, adding buffers to a congested network might even increase the amount of congestion since retransmission due to protocol time-outs consume more resource than consumed by retransmission caused by queue overflow. Therefore, one function of congestion control is to prevent throughput degradation associated with queue delay and buffer overflow.[1]

Once congestion occurs, a congestion control mechanism recovers by reducing a session's send rate. After the congestion has cleared, the mechanism allows a session to increase its rate. Each session should be given fair access to network resources.[2] The definition of fair, however, varies. For example, assume a router has a maximum switching capacity over an output link of 20 Mbps. Further assume that one source is capable of sending at a rate of 10 Mbps and another is capable of a 100 Mbps send rate. One definition of fair allocation is to allow each source to transfer at a rate of 10 Mbps. Another router, however, might allow each source to send at roughly 18% of its maximum rate. Not only is fairness difficult to define, it is also difficult to implement (especially in a connectionless network). Along with a router's queue service policy and packet drop policy, a network's congestion control mechanism is key to ensuring fair allocation of resources among competing sessions.

The throughput and fairness requirements of congestion control can be generalized by noting that the most visible function of any congestion control scheme is to ensure that a user obtains its desired quality of service (QoS).[3] A user's QoS might require bounded delays in addition to a guaranteed throughput. Other functions of congestion control include buffer management either at the end points or within the network and upholding policies imposed by the network. For example, the network might want 50% of available resources to be assigned to a particular protocol. Or, the network might allow a certain class of user to be able to allocate bandwidth in any amount while other users might be limited to a maximum amount of bandwidth.

---

[1] In the literature, flow control is sometimes differentiated from congestion control. For example, some authors view flow control as the mechanism that avoids sending data faster than a destination can absorb it while congestion control avoids or reacts to congestion within the network. Alternatively, some authors use flow control to describe both aspects of congestion. In this paper, we view flow control as one level of congestion control where a network's congestion control strategy consists of potentially multiple levels of control.

[2] This statement reflects a particular service discipline, namely a best effort service (such as what is provided by the Internet). An SNA network has a slightly different definition of fairness (based on priorities) and an Integrated Services network will have its own fairness schemes.

[3] The various components within a network that are associated with upholding a session's desired quality of service is typically referred to as traffic management. We view traffic management as a part of the network's congestion control strategy.

Figure 1 illustrates the effect of excessive load on a session's effective throughput [31]. The ideal throughput curve corresponds to perfect control which requires total knowledge of the network (obviously impossible). The controlled curve shows that there is an overhead involved with a control mechanism (e.g., additional control information that must flow, time delays to propagate control data). The uncontrolled curve shows the worst case: a total network collapse due to deadlock. All nodes hold on to buffers waiting for acknowledgments which never arrive (since there is not enough resource to successfully send and receive the acknowledgment). The more likely effect in modern packet switched networks is "congestion collapse" where the network enters a (stable) state where throughput is reduced to a small fraction of normal [61].

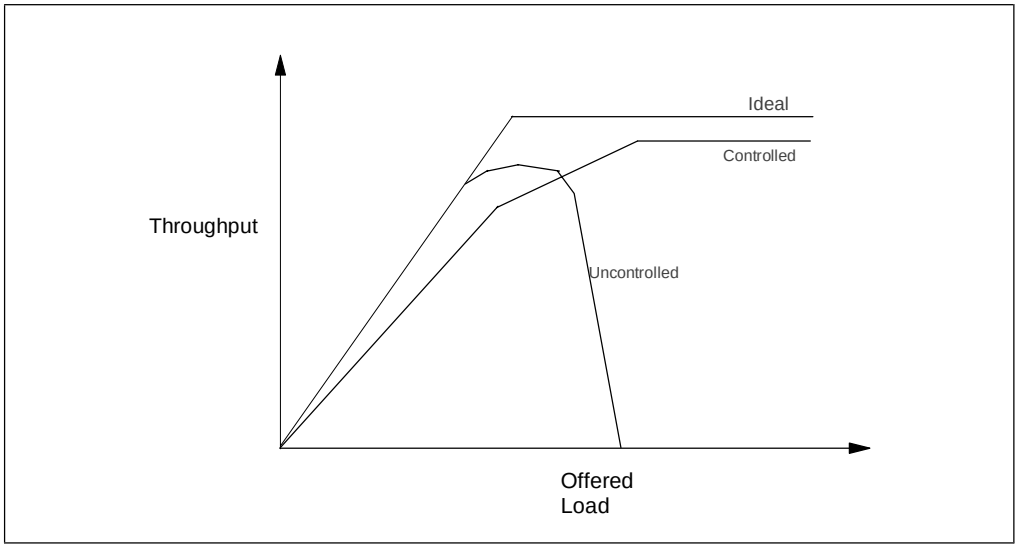

Figure 1

The challenge of congestion control is dealing with networks with large "pipes" (i.e., networks that have large bandwidth * delay products). A fiber based infrastructure, coupled with advances in networking technology (both hardware and software) have moved many networks from being bandwidth bound to delay bound. For example, consider a US coast-to-coast DS3 circuit (45Mbps). The transmission time of a 1500 byte packet is .267milliseconds while the one-way propagation delay is on the order of 30 mseconds. The "pipe" in this example is about 225 packets. Granted, the typical DS3 circuit is multiplexing hundreds of connections so the effective bandwidth available to a connection is much less. However, the point is that on closed-loop, large pipe connections, feedback might not arrive at the sender in time to prevent possibly significant packet loss.

The performance of a network is typically quantified by measures such as average throughput, delay variation, packet loss rates and reliability. Congestion control represents only one part of the network albeit a critical component when it comes to network performance. Network performance and consequently the selection of a network's congestion control mechanism is heavily influenced by the following aspects of the protocols driving the network:

- Connection mechanism: Whether the network is connection oriented or connectionless has significant influence on the design of the congestion control scheme.
- Routing algorithms: A routing algorithm that spreads load evenly over all possible paths helps reduce global congestion.[4]

[4]Actually, at best, optimal routing only delays the onset of congestion; routing can not eliminate congestion.

- Packet queueing policy and service policies at the routers: Fair queueing algorithms and packet loss algorithms (drop tail, random drop) have direct impact on throughput and session fairness.
- Round trip time measurements and retransmission time-out calculation: A low time-out will cause unneeded retransmission. On the other hand, a (too) large time-out leads to poor end-to-end throughput.
- Packet acknowledgment policy: Too many ACK's adds to congestion; too few ACK's affects the feedback delay for closed-loop control mechanisms (described below).
- Transport flow control scheme: The end-to-end flow control mechanism used at the transport obviously has a large effect on the dynamics of the network which in turn affects the selection of a congestion control scheme.

Even though network performance depends on the issues listed above (as well as many others), the importance of congestion control can not be understated. The congestion control mechanism in a network can quite literally make or break the network. In the next section we present a set of attributes associated with congestion control. While some of the attributes overlap, collectively they provide a framework that can be used to describe and differentiate the various control schemes.

1.2 Attributes of Congestion Control Schemes

Jain identifies two simple, high level classifications of congestion control schemes: those that dynamically increase resource as congestion occurs (e.g., bandwidth on demand) and those that dynamically decrease the demand [41]. It turns out that it is quite difficult to classify all forms of congestion control using a single level of classification. Instead, we identify the following a set of attributes (or descriptors) that provides a framework that we use throughout this report to describe specific congestion control schemes:

- Location in the network: different congestion control schemes operate at different points (or levels) in the network.
- Reactive or preventive: all congestion control schemes can be classified as either reactive (i.e., the control mechanism does not engage until after congestion occurs) or preventive (i.e., congestion is avoided).
- Open-loop or closed-loop (i.e., no feedback or with feedback) or a hybrid: all congestion control schemes will either operate with feedback or without.
- Window versus rate control: congestion control requires a mechanism to control the amount of data sent into a network. Window control implies packets are metered by receipt of credits from the receiver (or from the network). In rate control, packets are metered locally by the source at a constant rate.

There is clearly coupling between several of the attributes. For example, a reactive scheme is by definition a closed-loop mechanism. However, each attribute provides a slightly different view that collectively provides a thorough description of a congestion control scheme.

1.2.1 Location in the Network

The location of the control mechanism (also referred to as the level of the control) is a key attribute of a congestion control scheme. Figure 2 illustrates the different locations within a network where congestion control can be exercised [31].

- Hop level control
- Entry-to-exit level control
- Network level control
- End-to-end control

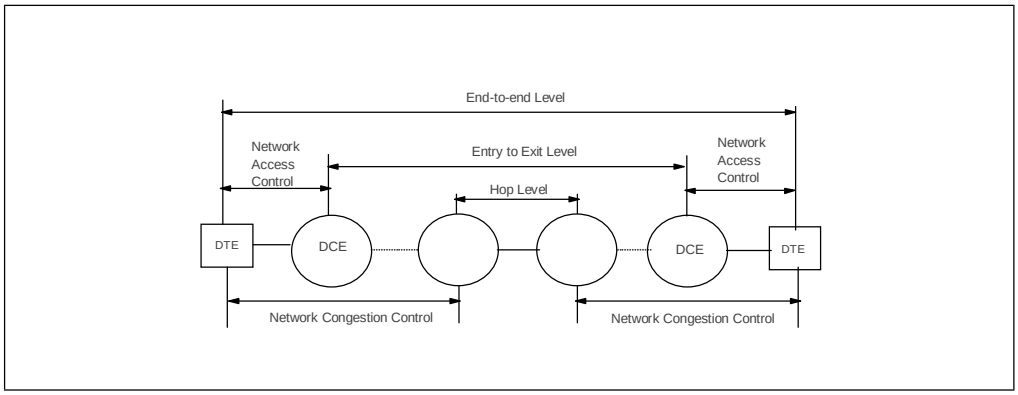

Figure 2

Hop level congestion control typically is an element of a data link control protocol. The objective is to prevent buffer congestion at the node. It is possible for a hop level congestion indication to make its way back to a traffic source using some "backpressure" technique. Hop level control can also be based on virtual circuits. Assuming a virtual circuit network architecture, a router might allow a maximum number of packets for each virtual circuit to be in transit. In a sense, it is a credit mechanism that is applied at each hop.

It has been asserted that hop-by-hop control is the only way to effectively control short term congestion (i.e., congestion that lasts less than a round trip time) [41]. If the backpressure makes its way back to the source, hop-by-hop control avoids the sometimes lengthy delay before end-to-end feedback propagates back to a source. Consequently, hop-by-hop rate-based congestion control scheme has been suggested for gigabit WAN's [58]. Hop-by-hop congestion control (at the virtual circuit level) is the basis for the credit congestion control scheme proposed by Kung for ATM's Available Bit Rate service [49]. However, primarily due to the requirements of a switch in a WAN environment, a rate-based, end-to-end feedback scheme was selected [7].

The objective of entry-to-exit congestion control is to prevent buffer congestion at the exit node caused by a source sending traffic at a higher rate than can be consumed by the exit node. A side effect of entry-to-exit congestion control might be to prevent global congestion. This will happen if the entry-to-exit control mechanism relies on feedback. As the network experiences congestion, the rate of the feedback returned to the entry node decreases causing the entry node to reduce its send rate. Entry-to-exit control might be utilized when the network and the end systems are under different administrative control.

The objective of network level control is to avoid network congestion via mechanisms located within the network and possibly at the end points. Network control differs from hop level and entry-to-exit control schemes since it attempts to control congestion globally rather than locally. There are two ways to implement network level control: network level congestion control and network access control. Network level congestion control requires feedback. If a router detects congestion, it will either implicitly or explicitly indicate congestion to a connection's endpoints (triggering the endpoints to react). The Internet community refers to this level of control as gateway congestion control [54].

Network access control will admit additional source traffic only if the network is not congested. An early example of network access congestion control is the Isarithmic scheme [6]. A network wide pool of "credits" exists that represents the available capacity in the network. Source traffic must have credits in order for the traffic to be admitted into the network. This scheme effectively limits the amount of traffic in the network. More modern implementations of network access control include traffic management functions such as admission control, bandwidth allocation and traffic shaping. While some believe that hop-by-hop level control is the only effective congestion control scheme for gigabit networks, others believe that network access control mechanisms are the only way to efficiently operate at gigabit speeds [3].

The objective of end-to-end control (referred to as flow control) is to prevent congestion at the end points (i.e., outside of the network). A network's congestion control scheme typically consists of multiple levels of control. For example TCP combines both end-to-end flow control along with a network congestion control scheme (slow-start and congestion avoidance [38]). SNA provides end-to-end flow control through session level pacing and congestion control through virtual route pacing [2, 40].

1.2.2 Reactive or Preventive Congestion Control

All congestion control schemes can be defined as either reactive or preventive. A reactive technique recognizes that the network is congested (typically by retransmit time-outs) and reacts by dropping the rate at which the session forwards packets into the network. Therefore, a reactive scheme requires feedback. A preventive scheme, on the other hand, tries to avoid congestion. Preventive congestion control can be either open-loop or closed-loop, while a reactive scheme is inherently closed-loop. In the next section, we describe in detail open-loop and closed-loop control.

A reactive/preventive description of congestion control is analogous to a description based on congestion recovery and congestion avoidance [88]. The goal of congestion recovery is to restore the network to its normal state after congestion has occurred. The goal of congestion avoidance is to keep the network either at or near the point of maximum power so that congestion will never occur (where power is defined as the throughput divided by the round trip time). A congestion avoidance scheme typically contains elements of congestion recovery.

1.2.3 Open/Closed Loop Congestion Control

A slightly more exact model of congestion control mechanisms than that provided by the other attributes discussed so far is in light of control theory. A control system is any system which exists for the purpose of regulating quantities in some controlled fashion [14]. Modeling congestion control in a network as a control system, while still an approximation with some error, does provide a good analogy. The quantity being regulated is obviously data, with the system composed of a network of queues and servers. Congestion control can be viewed as a control system for the purpose of maintaining the overall traffic within certain normal levels.

Figure 3a shows an open loop control system. The input, u(t) is in no way influenced by the output of the system, y(t). If the behavior of the system is not completely understood, then the output will not behave as precisely as expected. Figure 3b shows the closed-loop or feedback system. A feedback system is obviously better able to cope with unexpected disturbances and uncertainties about the system's dynamic behavior. It is not always true that a closed-loop system is superior to an open-loop system. For example, if the measured output has large errors, the closed-loop performance might be inferior to the open-loop system. However, in general, the advantage of closed loop control is its ability to dynamically adjust and maintain the steady-state performance via transient states.

The control theory analogy is used to identify and model congestion control schemes as either open or closed-loop. Yang and Reddy have proposed a taxonomy for congestion control algorithms based on control theory [88]. However, the control theory analogy can be taken a step further. It is possible to design and analyze congestion control algorithms using linear control theory. Some work has been done in this area, however this is a relatively new analysis technique [71].

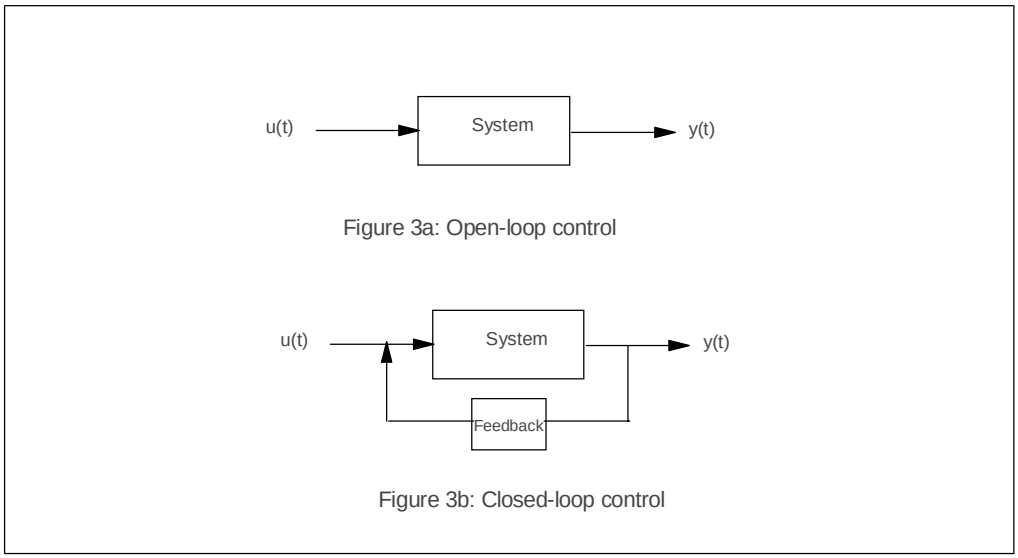

Figure 3

Open-loop schemes imply preventive congestion control with no feedback. Furthermore, "classic" open-loop control is rate-based and connection-oriented. Specifically, the components of such a system include:

- An admission control scheme: including the associated call setup and bandwidth allocation algorithms.
- A traffic shaping mechanism: given a negotiated rate, a sender utilizes a scheme such as leaky buck or token bucket to achieve the desired send rate (with the required flow characteristics) [81,64].
- A rate-based server algorithm: Each router must enforce each flow's send rate.

Bala and Sohraby have suggested that the only way to correctly handle congestion at gigabit speeds is through open-loop control for the following reasons [3]:

- Window based mechanisms rely on end-to-end feedback which is usually outdated by the time the message gets back to the source.
- Congestion control mechanisms must operate at the speed of the link. For this reason, computationally intensive hop-by-hop window based mechanisms are less desirable than simple schemes that can be implemented in hardware.
- The nature of future traffic (voice and video) will require some level of bandwidth guarantee. The source rate is determined by factors outside the control of the network.

The above factors suggest a simple open-loop scheme based on input rate control where the network controls streams of packets using the long term parameters of the rate control algorithm. A connection negotiates a rate (i.e., the connection specifies a set of rate control parameters). Due to the stochastic nature of traffic, there are times when a connection will exceed the negotiated parameters. The scheme

described in [3] will allow a sender to exceed its negotiated rate, however packets that represent the excess rate are tagged such that they are dropped first by the network in the event of congestion.

While it is true that there are scenarios where open-loop control is more effective than closed-loop control, in general however, open loop schemes are not robust enough to guard the network against all traffic patterns. For example, assume that a video stream requests enough bandwidth to support the maximum volume of data that the video source would ever generate (i.e., the maximum burst rate). In this case, network bandwidth would most likely be underutilized (since video data is inherently bursty). On the other hand, if the mean rate is requested, the connection might have to tolerate packet loss when the video source exceeds the mean rate (depending on if the network can support the excess throughput).

One of the tradeoffs associated with a closed-loop and open-loop discussion is network utilization versus quality of service (e.g., packet loss and delays bounds). An open-loop scheme that reserves resources at connection setup time is able to provide a guaranteed service to applications. However, the price paid for service guarantees are potentially poor network resource utilization. The predominant wide area networking protocols in use today (TCP/IP and SNA) clearly emphasize network utilization and consequently employ closed-loop congestion control schemes (along with a resulting best-effort service guarantees). One way for high speed networks (such as ATM) to address this tradeoff is to offer different services. For example, ATM offers a constant bit rate service (which is a classic open-loop scheme) primarily intended for real-time traffic that requires bounded delays. For data traffic (which requires packet loss guarantees but no delay guarantees), ATM provides an Available Bit Rate (ABR) service which essentially is a packet switched, statistical multiplexed scheme based on closed-loop congestion control.

Closed-loop congestion control schemes are differentiated by the nature of the feedback, by the frequency (and speed) of the feedback and by the response of the sender to the feedback. Reactive congestion control schemes imply a closed-loop system. However, the converse is not true. A preventive system can be either open or closed-loop.[5] The TCP slow-start congestion control scheme is an example of a closed-loop, reactive scheme[38].[6] TCP/Vegas [12], DECbit [70], Q-bit [72], and HPR[55] are all examples of closed-loop, preventive congestion control schemes.

In a closed-loop system, a sender adapts its load to the actual situation in the network by receiving some form of feedback. Ideally, the adaptation is made continuously. Figure 4 illustrates different aspects of congestion. The top curve is the classic throughput versus offered load curve. The point at which end-to-end effective throughput stops increasing represents the beginning of congestion. The middle curve shows that at this point, the round trip delay increases quickly (in fact, M/M/1 queueing delay increases exponentially). The objective is to keep the load near the “knee” of the delay/throughput curve (the upper plot) shown in Figure 4. The knee is the region of high throughput and low round trip delay and consequently represents the point of maximum power (as shown in the lower curve).

---

[5]Technically, a preventive control scheme is open-loop since a closed-loop preventive scheme detects the very early stages of congestion and then reduces source traffic in an attempt to avoid significant congestion. The advantage of closed-loop preventive control is that it avoids having to preallocate resources, enabling higher resource utilization through statistical multiplexing.

[6]In 1988, Van Jacobson proposed an improvement to TCP called congestion avoidance that is based on multiplicative rate decrease as time-outs occur). Congestion avoidance implies preventive flow control, however it can be debated if TCP’s congestion avoidance is really preventive. Even with the addition of the fast retransmit and fast recovery algorithms, the mechanisms are essentially reactive since they are engaged only after packet loss from buffer overflow occurs.

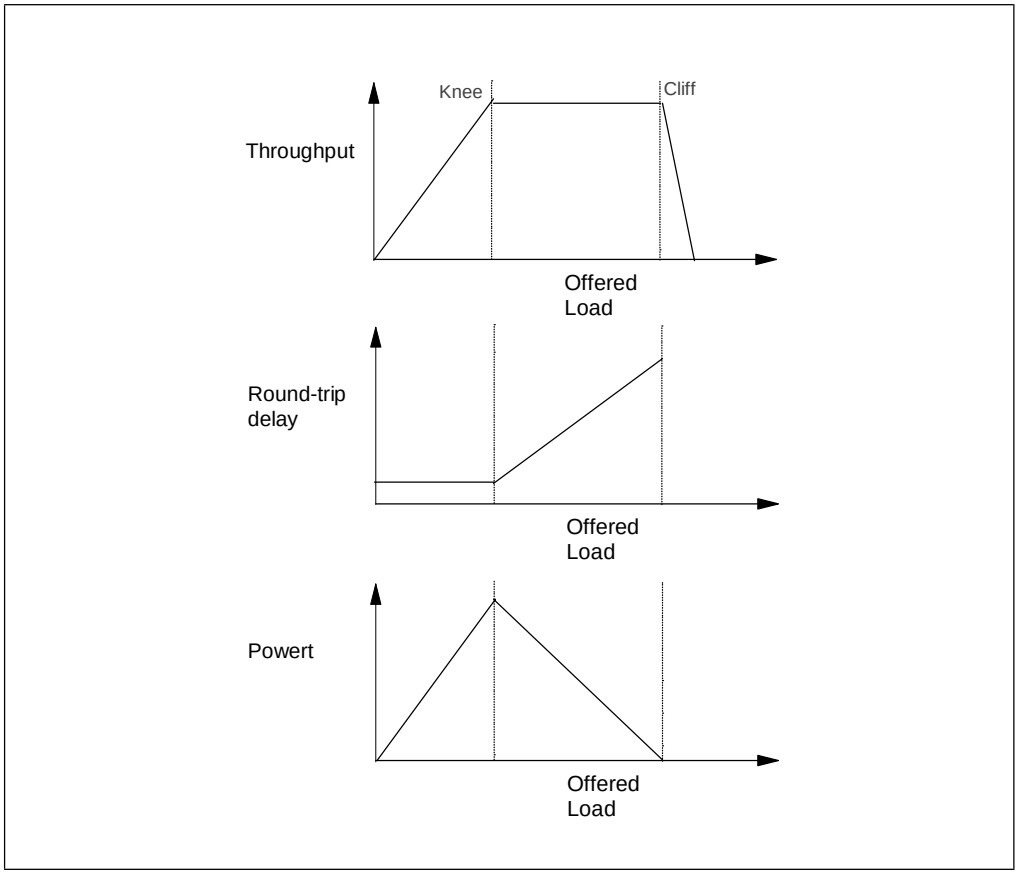

Figure 4

The concept of power was first introduced in [32] and analyzed in more detail by Kleinrock in [46,47]. A very useful observation made by Kleinrock is that an M/M/1 queueing system has maximum power when on the average there is only one message in the system. Kleinrock generalized this and showed that it also holds for M/G/1 systems. Therefore, in a multihop network, power is maximized when all links in the path are busy but the average queue depth at each node is 0. Several closed-loop, preventive control schemes (e.g., DECbit and Q-Bit) have been designed and optimized based on this observation.

A preventive, closed-loop control scheme tries to keep the network at the point of maximum power which is at a point around the "knee". Reactive control, on the other hand, will operate in the region around and beyond the "cliff". In the extreme, throughput drops quickly as most of the capacity is consumed by retransmissions.

The feedback in a closed-loop system is either implicit or explicit. Implicit feedback does not involve an explicit "send" or transmission of feedback signals. The feedback (such as a time-out or receipt of duplicate acknowledgments) is usually detected by the sender. TCP's slow-start and congestion avoidance algorithms are examples of control schemes that are based on implicit feedback. A key factor in the behavior of implicit control schemes is the packet drop policy used at the gateways. In the Internet, the predominant packet drop policy is drop-tail where a packet is dropped if it arrives at the gateway to find no available buffers. It has been shown that this algorithm, when combined with TCP, has shortcomings [34]. Floyd suggests that the Random Early Detect (i.e., RED) algorithm is more effective than drop-tail as it addresses several problems associated with TCP (which we will discuss later in the report) [28][7]

[7]Random Early Detection gateways support either implicit or explicit feedback where packets are either dropped or marked respectively. In this discussion, we assume an implicit feedback version of RED.

Explicit feedback involves an explicit send of feedback information. Explicit feedback can be characterized by the location of the source of the feedback, by the mechanism that transfers the feedback to the source and by the actual content of the feedback.[8] There are three possible sources of explicit feedback in a network:

- End-to-end feedback implies that the feedback information travels between the end points (i.e., the sender and receiver). Most transport protocols will use this technique as a form of flow control (i.e., the receiver can pace a sender if it is running out of buffers).
- Network feedback is generated by a router within the network when the router detects local congestion. As mentioned earlier, network congestion control is referred to by the Internet community as gateway congestion control. The feedback can be either forwards or backwards indications. The Source Quench facility of TCP/IP is an example of backward explicit congestion notification [11]. DECbit [70] and RED [28] are examples of forward network level congestion feedback.
- Local feedback implies that the feedback information comes only from a node's immediate neighbor. Examples of this include the hop-by-hop credit schemes proposed for ATM [49].

Examples of explicit feedback mechanisms (i.e., the way that a congestion indication is passed to a source) include:

- Congestion feedback via packets sent from routers to sources (e.g., source quench).
- Feedback included in routing messages exchanged among routers (e.g., tried in ARPANET's delay-sensitive routing [57]).
- End-to-end probe packets (e.g., HPR [55]).
- Congested routers fill in information in congestion feedback fields going in the reverse direction as the source packets (e.g., backwards explicit congestion notification as specified for Frame Relay in [15]).
- Congested routers fill in congestion feedback fields of packets in the forward direction (e.g., RED and DECbit).

The content of the feedback depends on if the data is a raw indication of congestion or if it is a rate change command generated by an entity other than the sender (i.e., the congested router itself or the receiver). Examples of specific feedback content includes:

- A binary indication of congestion (as observed at the router).
- An encoded value indicating a discrete level of congestion (perhaps using 2 bits))as observed by the router.
- A specific control message that informs a sender how to adjust its send rate. This message can be generated by the router itself or by the receiver. There are several levels:
  - A discrete rate adjustment command (e.g., decrease rate by 12.5%).
  - A continuous rate adjustment where a new send rate is specified.

Given that the Internet is (and will always be) a heterogeneous network environment, a widely researched class of congestion control algorithms are end-to-end, closed-loop, preventive schemes that do not require gateways to participate. Based on the taxonomy above, these algorithms use probe packets as the feedback mechanism (although some algorithms make use of existing control packets such as acknowledgment

[8]Explicit feedback can also be characterized by whether the indication comes from a router that maintain per flow state and issue feedback to particular flows that exceed their fair share [69]. Given the resource requirements at the router, these schemes have not been widely used.

packets). The content of the feedback will vary depending on the algorithm, however each scheme is similar in that they attempt to detect the onset of congestion by observing changes in network conditions (e.g., increased round trip times or changes in throughput). A likely evolution of TCP will be to integrate into the existing control mechanisms a congestion avoidance algorithm that is based on feedback (e.g., TCP/Vegas [12] which we present in section 2).

Control theory tells us that, for closed-loop systems, the control frequency should equal the feedback frequency. If the control is faster than the feedback, the system will oscillate and be unstable. On the other hand, if the control is slower than the feedback, the system will be slow to respond to change. Therefore, the frequency and responsiveness of the feedback in a closed-loop congestion control scheme is crucial to its effectiveness. A feedback scheme will not work properly if the congestion duration is less than the feedback delay. This implies that the congestion must be long-term (i.e., the congestion lasts for several round-trip delays). Because of this, closed-loop congestion control schemes can be ineffective in high bandwidth, large propagation delay networks. By the time the feedback arrives, the dynamics of the system might have changed significantly. Essentially, the feedback is out of date. Jain proposes that a solution to this problem is for such networks to offer a multi-level congestion control algorithm to handle both the short-term and the long-term congestion [41].

1.2.4 Window Versus Rate Control

Window or rate control (or perhaps a combination) represents the fundamental control mechanism in a network's congestion control scheme. The choice of control is completely separate from the other attributes of congestion control. For example, a rate control mechanism can be used in either an open-loop, preventive congestion control scheme or in a closed-loop, reactive scheme. Window control, which is the more common approach, imposes an upper bound on the amount of unacknowledged data that can be outstanding. It effectively dictates the largest amount of data that is either in the pipe or in the end point's queues (note this includes the source's transmit queue). Some window schemes combine congestion control with error recovery. For example, the data link protocol HDLC combines a Go-Back-N error recovery with a sliding window ARQ mechanism.

A window scheme can be implemented at any level in the network. Virtually all end-to-end flow control schemes (in production) are based on windows. A window congestion control scheme can be viewed as a preventive control scheme provided the upper window bound enforced by the scheme is appropriately picked. If the window value is incorrect (which is likely given that there is no way to accurately predict a connection's throughput and propagation delay over a wide area packet switched network) the network will be susceptible to congestion collapse.

While the basic window mechanism as described above is open-loop, the more common approach is a closed-loop window control scheme. A dynamic window scheme changes the meaning of the window variable from a fixed upper bound to the concept of a credit. For an end-to-end scheme, the receiving end of a connection controls the sender's send rate by changing the rate of credits returned back to the sender. A hop-by-hop credit-based scheme as defined in [49] is essentially a dynamic window scheme done at a lower level.

A key point is that a window control mechanism (or a rate control mechanism) is simply the mechanism that describes the metering process which indirectly defines the traffic shaping process. For window control, packets are triggered based on receipt of credits from the network or from the receiver. Since traffic shaping is determined by the credit arrival distribution, the distribution of packet departures from a window based sender is highly nondeterministic (i.e., uncontrolled). A rate control mechanism meters packets by generating credits at the source node at some constant rate. In rate control, traffic is shaped in any number of ways based on the requirements of the network.

It has been observed that window-based congestion control has significant problems when used over large bandwidth-delay paths. For example, a dynamic window mechanism (such as that used by TCP) can potentially allow a source to inject very large bursts of data into the network (e.g., if credits compress within the network and arrive at the source in a large burst). While a window mechanism can lead to uncontrolled bursts by a source, rate control is designed to control the burst rate at which a source injects packets into a network. There are many aspects to rate-based congestion control (e.g., open/closed loop, connection-oriented or connectionless, rate-based server algorithm at the routers), however the fundamental purpose of the rate control mechanism is to simply meter source packets into the network at a constant rate. At the end of Section 2 we explore rate-based congestion control in connectionless networks. We will see in Section 3 that rate control is a fundamental component of the Integrated Services Architecture. For now, the key point is simply to observe that an attribute of any congestion control scheme is whether the scheme uses window or rate control.

2 TCP/IP Congestion Control

TCP is an end-to-end transport protocol that provides reliable, in-order service. End-to-end flow control is integrated with a Go-Back-N error recovery mechanism. Network level congestion control is implemented via a reactive, closed-loop, dynamic window control scheme. The window normally increases by some amount each round trip time, however the window decreases when the sender observes congestion indications (e.g., packet loss). Ever increasing demands on the Internet have led to a number of incremental changes over the last 10 years designed to improve TCP/IP performance:

1. Improved round trip time measurement algorithm (Karn's algorithm) [44].
2. Slow-start and congestion avoidance [38].
3. Fast retransmit, fast recovery algorithms [77].
4. Improved operation over high speed, large delay networks [39].

A fundamental aspect of TCP is that it obeys a 'conservation of packets' principle where a new segment is not sent into the network until an old segment has left. TCP implements this strategy via a self-clocking mechanism: acknowledgments received by the sender are used to trigger the transmission of new segments. This self-clocking property is the key to TCP's congestion control strategy. If the receiver's acknowledgments arrive at the sender with the same spacing as the transmissions they acknowledge, and if the sender sends at the rate that acknowledgments are received, the sender will not overrun the bottleneck link.

Along with TCP's self-clocking admission control, other elements of TCP's congestion control include the congestion recovery algorithm (i.e., slow-start), the congestion avoidance algorithm, and the fast retransmit/recovery algorithms. The following is a very simplified description of these algorithms, see [77] for details.

Retransmit time-outs and a sequence of duplicate acknowledgments provide implicit indications of congestion in TCP. After a retransmission time-out, the TCP send rate drops to a single (maximum segment size) packet per round trip time. By the time a time-out occurs, the pipe is usually empty. Slow-start is used to get the acknowledgment metering process started. Slow-start allows TCP to quickly restore the send rate to approximately ½ the rate that existed before the time-out occurred by increasing the allowed send window (referred to as the congestion window or the *cwnd* value) by one segment each

time an acknowledgment is received. Once the window reaches ½ the value it had before the time-out (referred to as the slow-start threshold or the *ssthresh* value), the congestion avoidance phase begins where the congestion window increases much more cautiously (by one segment each round trip time).

The theory is that slow-start exponentially increases the send rate to an equilibrium point (i.e., a rate much lower than the rate at which packet loss was detected). Beyond this point, congestion avoidance probes the network for additional bandwidth. Early versions of TCP (prior to TCP/Tahoe) were limited to these two algorithms. The resulting behavior exhibited throughput oscillations as the TCP connection experienced repeated cycles (or epochs) of slow-start followed by congestion avoidance. A new epoch (i.e., a slow-start phase) begins after each retransmit time-out.

Fast retransmit attempts to avoid the retransmission time-out after packet loss occurs by predicting that a packet has been lost once three duplicate acknowledgments for the previous packet arrive at the sender. Immediately after fast retransmit, fast recovery attempts to minimize the throughput reduction that follows the retransmission. The goal is to prevent the pipe from going empty by allowing subsequent acknowledgments (that are received after a retransmission) to increase the *cwnd*. Going back to the 'conservation of packets' principle, each additional duplicate acknowledgment indicates that another packet (i.e., a packet sent after the lost packet) has left the network. As long as the send window is open, the sender can issue a new packet into the network.

The TCP congestion control algorithms are reactive and generally do not prevent congestion from occurring.[9] The algorithms try to obtain as much bandwidth as possible from the network by continually increasing the send rate until packet loss occurs. While the congestion control algorithms are effective in certain situations, there are other situations when TCP performance is poor. The following summarizes observed TCP behavior that collectively contributes to TCP's unpredictable performance.

Previous work has shown that one-way TCP connections tend to exhibit the following behavior [75,23]:

- Packet clustering: In TCP, packets that flow over a router link tend to be grouped by connection (i.e., sequential packets from one connection followed by a block of packets from another connection) rather than interleaved between different connections.
- Synchronized congestion window: TCP connections tend to synchronize such that they increase and decrease their *cwnd* value at the same time. In the extreme, the actual *cwnd* values of multiple TCP connections are identical.
- Synchronized packet loss: Synchronized connections tend to experience packet loss at (almost) the same time at a congested router.

The following other TCP dynamics and behavior have also been observed [29,85,91]:

- Poor performance when packet loss rates are high. For example the fast retransmit/recovery algorithms fail to avoid the retransmission time-out when multiple packets are lost during a round trip time or if the window size is low (less than 4 segments).
- Phase effects resulting from periodic traffic can lead to unfairness among competing TCP connections.
- Bias against bursty source traffic: For various reasons, a bursty TCP connection that requires on average a small amount of bandwidth, might consistently experience a disproportionate packet loss rate at a router compared to longer lived TCP connections.

---

[9]One can argue that the TCP congestion avoidance algorithm is truly designed to avoid congestion. However, unless the window sizes are optimally configured (i.e., the maximum congestion window is less than the queueing capacity of the network nodes and links), a TCP connection will eventually cause network congestion and therefore the control is reactive rather than preventive.

- Bias against TCP connections with longer round trip times. TCP increases its window each time an ACK arrives at the sender. Therefore connections with shorter path lengths will utilize a larger share of the available bandwidth since their window increases faster than other connection's with a longer round trip time.
- Acknowledgment compression contributes to bursty network conditions resulting in network instability. If a one-way TCP connection (e.g., an ftp file transfer) forwards a cluster of packets (i.e., back-to-back sends), into the network, as long as the network is not congested, the acknowledgment packets will arrive at the sender at the rate that the original packets traveled over the bottleneck link. However, if the ACKs experience congestion on the return path, they will accumulate (and compress the intervals between them) at the bottleneck router. When the router forwards the ACK packets, they arrive at the sender spaced by the transmission time of the small ACK packets. Since TCP uses ACKs to meter additional data into the network, a burst of ACKs that arrive at the sender at a high burst rate will cause the sender to burst additional source traffic into the network at a very high rate leading to congestion at the bottleneck link.
- TCP is also susceptible to bursty behavior after recovering from packet loss. If the recovery requires a time-out, once the time-out occurs, the pipe will likely be empty. Once the retransmitted packet arrives at the receiver, it can acknowledge not only the retransmitted packet but also all of the segments following the retransmitted segment which were sent to keep the pipe full. If, for example, the receiver acknowledges 10 packets, the sender will be allowed to send possibly up to 11 packets.

A crucial question is if these behaviors exist in real networks. TCP's bias against bursty traffic and against connections with long round trip times are inherent to the TCP protocol. It is not difficult to show that these problems exist in any TCP network. However, there is debate in the literature if behaviors such as synchronization effects and ACK compression are real problems.[10] For example, Floyd suggests that it is unlikely that the dramatic phase effects observed in [29] are significant in real systems since synchronized behavior disappears as the amount of randomness in the system increases.[11]

While the majority of TCP analysis has been simulation based, there have been several empirical studies performed that illustrate that TCP can exhibit unwanted behaviors such as synchronization effects and ACK compression. For example, Mogul indicates that ACK compression has been observed in networks [60]. Additionally, work done in [82] points out that periods of high packet loss (at times over 30%) have been observed in certain regions of the ASFNET when average link utilization's were in the 50% range. The conjecture is that this is due to the synchronization of a large number of TCP connections.

To what extent real TCP networks exhibit unwanted behavior such as synchronized connections and ACK compression is difficult to assess. On the positive side, many of the problems associated with TCP either have been or are being addressed. Enhancements such as TCP/SACK [21], Random Early Detect [28] and TCP/Vegas [12] can provide significant improvements to TCP. Collectively, these enhancements represent three areas that have received much attention by the Internet research community: improvements to TCP in the event of multiple packet loss, enhanced gateway congestion control schemes and improved congestion avoidance algorithms (without assistance from gateways). A fourth area that we explore are the issues associated with rate control (as opposed to window control) in a TCP/IP environment. The remainder of this section surveys the issues associated with each topic.

---

[10]Network administrators typically monitor performance and can tune the network to minimize the effects of unwanted TCP behavior. However, this is time consuming and can result in poor resource utilization.

[11]A clear conclusion is that care should be used when drawing conclusions based solely on simulation results.

2.1 Improved Behavior in Response to Packet Loss

The throughput loss resulting from TCP's reaction to a single dropped packet is minimal. However, when multiple packets are dropped within a single round trip time, the throughput reduction is significant. The throughput loss increases proportionally with the product of the bandwidth and the propagation delay.

Due to global synchronization effects in TCP and also due to the bursty nature of network traffic, multiple packet loss is a common occurrence in TCP/IP networks. Paxson has observed that 13% of TCP connections in the Internet exhibited occurrences of multiple packet loss that required a retransmission time-out (based on a sample of 2299 TCP connection traces) [21]. Another cause for multiple packet loss is the initial slow-start increase of TCP. When a TCP connection starts, the *ssthresh* is typically set to the maximum window size (e.g., 65536 bytes) which means that a new TCP connection will typically increase its rate exponentially until packet loss occurs. After this, the *ssthresh* is set to ½ the value of the send rate of the sender when it first deduces that a packet is lost. This provides a setting for the equilibrium point allowing the next slow-start cycle to be better behaved.

Hoe has proposed several improvements to TCP's congestion control schemes to address the performance problems associated with multiple packet loss [36]. There are two major improvements. First, she proposes a better initial value for *ssthresh*. The obvious benefit is to avoid the uncontrolled increase of a new TCP connection's send rate. The challenge is how to do this.

One technique is to use the bandwidth-delay product. The bandwidth can be estimated by observing the interarrival time between back-to-back acknowledgments. Based on the Packet-Pair concept [45], if two data packets are sent back-to-back (and arrive at the receiver back-to-back), the acknowledgments should arrive at the sender separated by the transmission time of the bottleneck link (this assumes ACKs are generated for every packet). From this, the bottleneck link capacity can be estimated. The computed bandwidth-delay (where the delay is obviously the round trip time) can be set to the threshold (i.e., the , *ssthresh* value).

The second improvement proposed by Hoe is to improve the fast retransmit/recovery algorithm by making use of additional information implicitly provided by duplicate ACKs that arrive after a fast retransmit. Let's say that packets 1 through 7 are sent but only packets 1,3 and 5-7 arrive at the receiver as illustrated in Figure 5a. Upon receipt of packets 3 and 5-7, the receiver will send the duplicate ACK 3. When the third duplicate ACK arrives at the sender, it will retransmit packet 2. When the receiver receives packet 2, it acknowledges packet 3 but not any of the other packets that it has received. Since there are no more packets in transit for the receiver to acknowledge, the sender stalls and requires a time-out to recover. Hoe's suggestion is for the sender to observe that once the acknowledgment for packet 3 arrives (after packet 2 is retransmitted), this is a clear indication that packet 4 has also been lost. Therefore, upon receipt of the acknowledgment for packet 3, the sender should immediately retransmit packet 4. Effectively, the fast recovery process is extended until the highest segment number that was sent has been acknowledged. Figure 5b shows that in this case, a time-out is avoided.

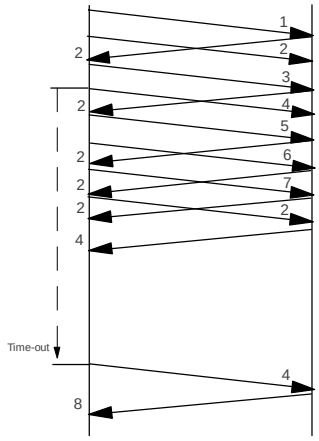

Figure 5a. TCP/Reno with multiple packet loss

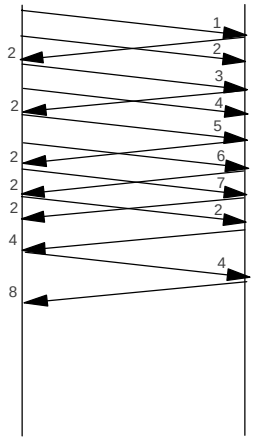

Figure 5b. Enhanced TCP/Reno with multiple packet loss

Figure 5

Another approach to improving TCP performance during periods of high packet loss is to enhance TCP's error recovery mechanism with selective acknowledgment and selective repeat capabilities [21]. A selective acknowledgment scheme was first proposed by Van Jacobson in 1988 [10]. Fall and Floyd recently have championed Jacobson's original proposal (with a simple enhancement) in the IETF resulting in the ratification of RFC 2018 [56]. The RFC only defines the mechanism by which a receiver can inform a sender of blocks of data that have been received, it does not specify how the sender should react with the information. We anticipate that within the next several years, an RFC will be generated specifying the sender's congestion control response to a TCP/SACK acknowledgment that indicates multiple packet loss.[12]

---

[12]Advances in TCP congestion control are painfully slow. First a problem is observed and solutions are studied (and studied). Prototypes are developed and measurements over real networks are obtained to verify the theoretical results. The first official stamp of approval occurs once the enhancement is implemented and distributed in a release of BSD UNIX. The last significant TCP release occurred with version 4.3 BSD when TCP/Reno was introduced. The final step is for an RFC to be generated that specifies in detail the enhancement. As an example, the fast recovery algorithm was introduced in 1991 by Van Jacobson in an informal note to the Internet community. It was added to 4.3 BSD Reno and only recently documented in an RFC (RFC 2002 [78]).

Fall and Floyd have implemented TCP/SACK in the ns simulator [24]. The TCP/SACK RFC specifies a SACK options field that holds reports of noncontiguous data that have been received and queued. Fall and Floyd purposely made minimal changes to the current Reno control algorithms- in fact the goal was to preserve the algorithms. The biggest improvement offered by TCP/SACK (over the base TCP/Reno or Hoe's proposal) is that during periods of high packet loss, TCP/SACK recovers much more quickly. TCP/Reno (and Hoe's algorithm) can at most recover only 1 lost packet per round trip time. TCP/SACK allows TCP to treat an occurrence of multiple packet loss as a single indication of congestion rather than as multiple separate indications (and a separate recovery phase required for each indication). Preliminary measurements of a TCP/SACK implementation (which is loosely based on Floyd's ns implementation) shows throughput improvements up to 50% over long (16-18 hops) paths [4]. However, under different network conditions there clearly might not be any improvement.

Mathis and Mahdavi have extended the base TCP/SACK algorithm by decoupling the congestion control (i.e., the fast retransmit/recovery algorithms) from error recovery. The goal of their algorithm is to perform precise congestion control during recovery by keeping an accurate estimate of the amount of data in the pipe. The algorithm, known as the Forward Acknowledgment congestion control algorithm (FACK) uses the additional information provided by the SACK option to keep an explicit measure of the total number of bytes outstanding in the network. Rather than rely on duplicate acknowledgments to indicate that a segment has left the network, the FACK sender calculates the exact number of packets that are in the pipe based on what has been sent and based on the SACK information that indicates what the receiver has successfully received. The fast retransmit algorithm is modified to make use of this information.

The issues associated with improving performance during periods of high packet loss can be summarized as follows:

- The main behavior of slow-start and congestion avoidance can not change. TCP must drop to slow-start in the event of a time-out and TCP must drop to congestion avoidance in the event of a retransmission due to fast retransmit algorithm.
- An area of improvement is to eliminate unnecessary retransmissions (i.e., by adding a selective repeat protocol).
- Another area of improvement is for the sender to more accurately track the pipe size after a retransmission. Congestion avoidance dictates that, after a fast retransmission, the pipe must drain by ½. The challenge is to not let the pipe drain more than this.

2.2 Survey of Gateway Congestion Control

As mentioned earlier, gateway congestion control is a form of network level congestion control where the gateway provides either explicit or implicit feedback indications [54]. The study of gateway congestion control has primarily been performed in the context of TCP/IP. Gateway congestion control schemes can be differentiated in one of several ways. Some schemes require the gateway to maintain state on each flow while others do not. Another differentiator between gateway congestion control schemes is whether implicit or explicit feedback is issued by the gateway. In this section, we focus on connectionless gateway congestion control and specifically on the source quench, DECbit and Random Early Detect algorithms.

The simplest form of gateway congestion control in an IP network is a drop tail packet drop policy. TCP uses packet loss as an implicit congestion indication and reacts by cutting its send rate. RFC-792 [67] specifies a more controversial form of gateway congestion control in TCP/IP networks known as source quench. When a gateway responds to congestion (by dropping packets), it may send an ICMP

source quench message to the source of the dropped packet. The Gateway Requirements RFC, RFC-1009 [11], specifies that gateways should limit the frequency of source quench messages to reduce the additional traffic. A significant problem with the scheme is that neither RFC specifies when a gateway should issue a source quench message. Furthermore, the action of the host when it receives a source quench message is not specified. When a TCP/Reno sender receives a source quench, it waits until the current retransmit timer expires and then enters slow-start [77]. Given these problems, it has been observed that source quench is rarely used [25].

In 1986, the Internet experienced a series of 'congestion collapses' where throughput was significantly reduced due to severe congestion. This led Van Jacobson to develop and introduce a set of improvements to TCP's congestion control that included slow-start and congestion avoidance [38]. At this time, Ramakrishnan and Jain took a different approach to solving the same problem. They proposed a scheme for congestion avoidance that used explicit feedback from a gateway [70]. Originally referred to as Explicit Binary Feedback, it is generally referred to as DECbit since it was intended primarily for DEC's Digital Network Architecture (DNA) protocol. However, the scheme can be applied to any connectionless network that uses a virtual circuit oriented transport protocol (such as TCP).

In brief, when a DECbit router detects congestion, it sets a congestion indication bit in the network header of a data packet that is flowing in the forward direction. When the data packet reaches the destination, the congestion indication bit is copied into the transport layer header of the acknowledgment packet and forwarded back to the sender (who performs the rate adjustment by modifying the window value). DECbit differs from the source quench scheme in two ways:

- DECbit is a forward congestion notification (FECN) rather than a backwards notification scheme.
- DECbit is a congestion avoidance scheme rather than a congestion recovery scheme.

It is interesting to understand the key design aspects of the DECbit algorithm. The algorithm at the router has two main policies. The first policy defines when a router should set the congestion indication bit. Clearly the best indication of congestion at a router is queue levels. The algorithm is designed to maximize power. As mentioned earlier, maximum power is achieved when on average there is only 1 packet in the system. Therefore, a router will set a congestion indication bit in packets that flow over an output link once the average queue level (including the packet in service) at the link exceeds 1.

The second policy of a DECbit router implements a filter on the feedback. The average queue length is updated by the router upon each packet arrival. A filtering function determines a meaningful time interval used for the queue level average. If the average is calculated periodically based on fixed time intervals, then the accuracy of the sample is dependent on line speeds. For example, clearly more frequent queue samples are required over a 45 mbps link than over a 64kbps link (a queue sample is when the queue average over a time interval is calculated). A better approach is an adaptive averaging technique such that the frequency of samples dynamically change based on an 'on/off' cycle of busy periods.[13] An on/off cycle is defined as the busy+idle interval as seen by the router (i.e., the time between busy periods). To account for potentially long cycles (i.e., a relatively long interval of time where there is at least 1 packet in the router), a sample is computed based on the queue average during the previous cycle as well as the current. Therefore, when a packet arrives at a router, the router calculates the average queue length for the last (busy+idle) period plus the current busy period.

At the source, the key aspects of the DECbit algorithm include:

- The decision frequency: the source adjusts its rate (i.e., alter its window size) each time that approximately two windows of data have been acknowledged (i.e., every 2 round trip times). The rational is that in a computer network, it takes one round-trip time to affect the control. It takes an additional round-trip time to observe the effects of the resulting change. Therefore, a source rate should be adjusted every other round-trip time.

[13] A busy period refers to when a server in a queueing system is busy.

- Use of received information: the main point here is that only the congestion bits from the packets received from the last window of data are used in the rate adjustment decision.
- Signal filtering: the source will decide to decrease the rate only if more than 50% of the packets received have the congestion indication bit set. In order to maximize power (i.e., where the router assumes that a link is congested if on average there is more than 1 packet queued over some time interval), Ramakrishnan and Jain show that the optimal filter at the source is 50%. Therefore, the source assumes the network is congested if more than 50% of the acknowledgments received during the last window have the congestion indication bit set.
- Increase/decrease algorithm: an additive increase and multiplicative decrease algorithm is used. If the rate decision indicates that a rate increase is required, the window is increased by 1. If the rate is to decrease, the window is decreased by 12.5%. Extensive research has gone into finding the correct combination of increase/decrease factors [18]. The issues include assuring fairness across multiple sources (and related is the time to achieve fairness) and minimizing oscillations of source sending rates. The goals are actually conflicting. A small decrease factor increases the time for connections to converge to a fair share of available bandwidth. However, large decrease factors minimize oscillations. It turns out that the increase/decrease parameters chosen for DECbit (increase by 1, decrease by 12.5%) were selected specifically to minimize oscillations. Furthermore, the decrease factor of 12.5% was selected since it can be implemented using shifts thereby avoiding floating point calculations.

It turns out that DECbit is prone to oscillations, exhibits bias against bursty traffic and has other unfairness problems [72,16]. While it is possible to apply the DECbit algorithm to TCP/IP, it is questionable as to how well even an improved version of DECbit would perform in a TCP/IP environment. Floyd and Jacobson took another approach and designed a congestion avoidance scheme specifically for TCP. The Random Early Detection (i.e., RED) algorithm is functionally similar to DECbit in that it is a gateway-based congestion avoidance scheme based on forward explicit congestion indications. As with DECbit, a RED gateway monitors an average queue level looking for the beginnings of congestion. Once the router determines that a link is congested, it will mark packets that flow over the congested link (RED also supports a mode where packets are dropped instead of marked).

However, the RED algorithms differ from DECbit in several aspects. First, each algorithm utilizes a different method to calculate the average queue size. Second, each algorithm has its own method of selecting which connections to notify of congestion. In the following discussion of the RED algorithm, packet marking can be replaced by packet dropping.

The RED algorithm is summarized as follows:

for each packet arrival
calculate the average queue size *avg*
if *min* < *avg* < *max*
calculate marking probability *P*
with probability *P*, mark (or drop) the arriving packet
else if *max* < *avg*
mark (or drop) the arriving packet

The average queue size is calculated based on an exponential weighted moving average[14]:

$$avg = (1\text{-}w)avg + wq$$

The *w* is a weight that determines the tolerance to short-term congestion. The *q* value is the current queue size observed when the queue size average is computed (i.e., when a packet arrives). Floyd and

[14]The algorithm as presented here is in terms of packets (i.e., the *avg* is the average number of packets queued over a given time interval). Floyd and Jacobson show a simple modification to make the algorithm work in units of bytes.

Jacobson use a value of .002 for *w*. The *min* and *max* determines the operating range for the queue (in units of packets). While these values are configurable, Floyd and Jacobson recommend that the *min* be larger than 1 and the *max* be at least twice the *min* value. The optimal values are dependent on the characteristics of the traffic. For example, if the source traffic is bursty, then the *min* should be large to ensure high link utilization (of course this assumes that the router has sufficient queue capacity). For the simulations performed in [28], the *min* value ranged from 3 to 50 packets and the *max* was set to *3min* packets.

Compared to DECbit, the RED average queue size is typically calculated using longer time intervals. The average queue level will also tend to be higher than a DECbit router (primarily due to the fact that DECbit assumes an average queue size of 1 implies congestion while RED uses a larger threshold). The difference in behavior is essentially a reflection of the differences in the dynamics and characteristics between a DNA network and a TCP/IP network.

The mark (or drop) probability is based on the following formula:

$$P = max\text{-}prob * (avg - min) / (max - min)$$

The *max-prob* gives the maximum packet-marking probability that is used when the average queue size reaches the maximum threshold. A typical value for *max-prob* is 1/50 which says that once an output link at a router becomes congested, roughly 1 out of 50 packets are marked (or dropped). Given the mark (or drop) probability, a uniform random variable is generated that determines the number of packets that are processed before the next packet is marked (or dropped). Combined with a FIFO queueing discipline, the marking algorithm has the following goal: the probability that a connection is notified of congestion is proportional to that connection's share of the bandwidth. When compared to DECbit, the RED scheme will most likely mark fewer packets. This saves considerable overhead at the router, allowing the packet forwarding process to remain essentially untouched by the algorithm if the queue average and the marking probability calculations are done in parallel.

As mentioned, a RED gateways can chose to drop packets instead of marking packets. A TCP/IP network can benefit from an implicit feedback version of the RED algorithm in the following ways:

- Bounds queue levels helping to control packet delays.
- Eliminates TCP's global synchronization effects by fairly distributing congestion indications across all TCP connections.
- Eliminates TCP's bias against bursty traffic.
- Can improve performance by reducing the frequency of multiple packet loss experienced by a connection during a round trip time.

While RED provides significant advantages when used in an implicit mode, the benefits increase substantially when used with explicit congestion notification (ECN) [25]. When operating in an environment where all TCP sources/destinations are cooperative (i.e., all routers participate in the RED algorithm), the RED gateway sets a bit in the IP header (i.e., marking the packet) based on the algorithm described above. The TCP receiver observes that an arriving packet is marked and sets a similar bit in the IP header of the ACK packet. When the source receives an ACK with a congestion indication, it must respond (described below) by reducing its send rate. Unlike DECbit where the source needs to filter the indication signal, the filtering is done at the gateway and so the source can react immediately to a single congestion indication.

The advantages of explicit TCP congestion avoidance includes the following[15]:

- Explicit congestion notification reduces unnecessary packet loss which increases throughput and decreases average delays experienced by packets.

[15]The advantages of a modified TCP sender/receiver that participate in RED's ECN would also benefit from the randomizing effects of the algorithm resulting in less susceptibility to global synchronization and less bias against bursty traffic sources.

- Sources are informed of congestion quickly and unambiguously.
- Allows subnetwork congestion indications to be utilized effectively (e.g., frame relay's FECN signals and ATM's ABR feedback indications).

When a TCP source receives an explicit congestion indication (ECN), Floyd proposes the following guidelines:

- TCP's response to an ECN should be similar over longer time scales to its response to dropped packets or to time-outs.
- Over smaller time scales, TCP's response to ECN can be less conservative than its response to lost packets (since an ECN is not an indication that a packet has been lost, but rather that the queue at the router has reached some threshold level).
- The receipt of a single ECN should trigger a response to congestion (which is unlike the DECbit scheme).
- TCP should react to ECN at most once per round trip time.
- TCP should continue to use incoming ACKs to meter new packets into the network. Additionally, TCP should not alter the basic slow-start and congestion avoidance algorithms.

Floyd coded the changes to the TCP sender as follows: when an ECN is received and if no other indications have been received for at least one round trip time, the *cwnd* and *ssthresh* parameters are reduced by ½ and the sender stays in congestion avoidance. If three duplicate ACK are received within one round trip time after the receipt of an ECN message, the source does not reduce its *ssthresh* or *cwnd* (since it already reduced it). However, the sender retransmits the packet and then transitions to fast recovery where incoming duplicate ACKs clock additional outgoing packets. Based on simulation results, the RED ECN scheme can significantly improve the stability and performance of a TCP/IP network. It is interesting to note that Floyd stresses that the benefit of ECN is not necessarily increased connection throughput but rather lower average packet delay (with less variance) for low-bandwidth, delay-sensitive TCP connections.

2.3 End-to-End Congestion Avoidance Schemes Based on Windows

As mentioned, there exists a class of congestion avoidance algorithms that are end-to-end such that the endpoint detects incipient congestion based on observed traffic dynamics. While TCP's network level congestion control algorithms are reactive (based on a simple drop-tail gateway congestion control policy), the schemes discussed in this section are preventive. Furthermore, they are implemented at the endpoints and are designed to avoid packet loss thereby avoiding the need for a gateway congestion control packet drop algorithm. In this section, we focus on such schemes that are window based schemes. There is a set of similar rate-based schemes that we discuss at the end of this section. We briefly describe the following window based, end-to-end congestion avoidance schemes:

- Jain's Congestion Avoidance using Round-trip Delay (CARD) [40].
- Mitra's Dynamic Windows algorithm [59].
- Crowcroft and Wang's Tri-S scheme [83].
- Wang's Dual scheme [84].
- Brackmo and Peterson's Vegas scheme [12].

As the name implies, Jain's Congestion Avoidance using Round-trip Delay (known as CARD) scheme requires a sender to adjust its send window based on observed changes in round trip times. Similar to

DECbit, CARD defines an additive increase and multiplicative decrease rate adjustment algorithm. The key difference is implicit feedback (changes in round trip times) versus explicit feedback. CARD represents at best an introduction to the issues associated with end-to-end congestion avoidance schemes. While CARD as described in [59] was never implemented (or at least as far as we are aware of), it is important as it lays the groundwork for other related work. Jain suggests the following benefits provided by CARD:

- The scheme can be used over a heterogeneous network where one subnetwork might offer one form of explicit congestion indications while another subnetwork might not support any.
- The scheme requires no network overhead and requires no additional probe packets.
- The scheme requires no change to packet headers.

As with DECbit, CARD uses the concept of power as its optimality criterion. Jain initially developed CARD based on a deterministic network model. The decision to increase or decrease the window is determined by a relation called the network delay gradient (NDG) that is a function of two sequential round trip time samples and the respective window size . Specifically, the window is decreased by 1/8 if the following product is positive:

$$((W_{current} + W_{old})/(W_{current} - W_{old})) * ((RTT_{current} - RTT_{old})/(RTT_{current} + RTT_{old}))$$

$RTT_{current}$ and RTT_{old} represent the current and previous measured round trip time sample when the window values are $W_{current}$ and W_{old} respectively. The window is increased linearly (by 1 packet) if the product is less than or equal to 0. Jain shows that the NDG calculation leads to a socially optimal rate control decision (meaning that the scheme is fair). As in DECbit, the window is adjusted every other round-trip time.

Jain makes the following observations and suggestions for future study:

- To be a viable scheme, it needs to be adjusted to work in a probabilistic network (i.e., where the service times at a router vary depending on packet size and internal delays). For example, multiple samples of delay can be taken and the mean and confidence interval can be calculated and used to generate the NDG value.
- If the minimum delay is known (either approximated as the bottleneck link transmission time or the sum of transmission times over each hop), then an even more ‘socially optimal’ rate decision can be made.
- It has been observed that it is not possible to optimize network power using a decentralized algorithm [43]. Jain suggests that a different interpretation of power might be more optimal.

Mitra’s dynamic windows algorithm adapts the window based on changes in round trip delays. The algorithm is based on an asymptotic analysis of queueing network models where it can be shown that optimal steady state performance is related to the mean round trip time through an explicit relation known as a design equation. Every packet is sampled (i.e., when applied to TCP, the TCP timestamp option is required along with an accurate timestamp rather than a coarse grained timer tick count as specified by the timestamp option) and the delay is fed into the design equation which indicates if the send rate should increase (by 1 packet), decrease (by 1 packet) or stay the same.

It is questionable how well CARD or Dynamic Windows would perform when applied to TCP. Wang and Crowcroft have proposed two different end-to-end congestion avoidance algorithms designed specifically to work with TCP: Tri-S [83] and DUAL[84]. The Slow-start and Search (referred to as Tri-S) algorithm has several fundamental design points. First, rather than continuous cycles of additive increase and multiplicative decrease, Tri-S attempts to quickly establish an optimal and fair operating point each time when there are major traffic changes. Traffic load is deduced by using a metric called the normalized throughput gradient (NTG). A throughput gradient is defined as:

$$TG(W_n) = (T(W_n) - T(W_{n-1}))/(W_n - W_{n-1})$$

The W_{n-1} and W_n represent two sequential window sizes. $T(W_n)$ is the throughput when the window is W_n. Throughput is defined as

$$T(W_n) = W_n/D_n$$

where W_n is the number of bytes acknowledged by the time the ACK for the n'th packet arrives. D_n is the round trip delay when the ACK of the n'th packet is received. The normalized throughput gradient is defined as:

$$NTG(W_n) = TG(W_n)/TG(W_1)$$

As traffic load increases, the $TG(W_n)$ decreases to 0. The $NTG(W_n)$ varies approximately in the range [1,0]. Under light traffic, the $NTG(W_n)$ is 1. The $NTG(W_n)$ decreases gradually as the load increases reaching about 0 when the path is saturated.

When a connection starts, Tri-S essentially enters a slow-start phase (called the initialization phase) where the window size is increased by one packet each time an ACK is received. Once a maximum window is reached, Tri-S transitions to congestion avoidance (called increase mode) where the rate increase slows to a linear increase rate. During increase mode, once each round trip time the $NTG(W_n)$ is checked. If the value is greater than a threshold value (NTG_t), the window is increased (by the fraction $1/W_{current}$). If the $NTG(W_n)$ is less than the threshold , the window size is decreased by one packet. Otherwise, do nothing. The simulation results presented in [83] (which sets NTG_t to ½) shows that the scheme is able to reduce the oscillations that are associated with TCP.

The scheme is highly dependent on the initial round trip delay value (D_1). The scheme also relies on synchronized packet loss to ensure fair allocation of bandwidth for competing connections (as does TCP). If all connections drop to slow-start at the same time, all the connections will get about the same amount of bandwidth. However, as observed in TCP, this behavior is unfair to longer path length connections.

Wang and Crowcroft's other algorithm, known as DUAL (for dual adjustment scheme), monitors the round trip times and overrides the slow-start and congestion avoidance window increase with a window decrease if the measured round trip time increases beyond a threshold [84]. DUAL differs from Tri-S in the following ways:

- Rather than using throughput, congestion is monitored by observing changes in round trip times.
- Rather than an additive increase and additive decrease rate adjustment algorithm, DUAL uses the DECbit approach of additive increase and multiplicative decrease.

DUAL assumes that the round trip delay of the n'th packet is defined as:

$$D(n) = D_p + D_q(Q_n)$$

The D_p term represents the round trip propagation delay and transmission times at each hop. The $D_q(Q_n)$ term represents queueing delay. The algorithm further defines the minimum and maximum possible delays to be:

$$D_{\min} = D(0) = D_p$$
$$D_{\max} = D(Q_{\max})$$

where $Q_{\max}$ is the maximum queue length at the bottleneck link. DUAL approximates these values based on the minimum and maximum measured round trip times:

$$D_{\min} = \min(measured\_rtt, rtt\_\min);$$
$$D_{\max} = \max(measured\_rtt, rtt\_\max);$$

Over the life of the connection, clearly the $D_{\min}$ variable approaches the round trip time when there are no queueing delays and $D_{\max}$ approaches the round trip time when the bottleneck link buffer is full. DUAL defines a threshold round trip time that determines the operating point:

$$D_i = (1 - \alpha)D_{\min} + \alpha * D_{\max}$$

The parameter α is used to set the operating point. Once a sampled rtt exceeds the threshold value, the window is reduced by 1/8. The simulations done in [84] chose a value of ½ for α which corresponds to a rate reduction when the bottleneck link queue is ½ full.

As with Tri-S, DUAL has some fundamental problems. The performance of the scheme depends on accurate D_{min} and D_{max} values which are difficult to obtain in an Internet environment. It is likely that Dual will exceed the target threshold queue level at the bottleneck link when there are many connections (due primarily to persistent congestion).

TCP/Vegas is the most recent and the most widely accepted end-to-end congestion avoidance scheme [12]. TCP/Vegas offers three changes. The first change is fairly minor, while the other two changes represent the major contribution: an end-to-end congestion avoidance scheme (referred to as CAM for congestion avoidance mechanism) that requires only a change to the TCP send algorithm (i.e., it will work with all existing TCP receivers).

The rational behind the first Vegas change is that time-outs in TCP are extremely costly, especially on LAN's. The problem is primarily due to the reliance on a coarse grained timer tick (500ms) to implement retransmission time-outs. Consequently, Brackmo and Peterson propose the following change:

- Maintain a fine grained retransmission time-out that runs in parallel to the standard coarse grained timer.
- When either of the following events occur, check to see if the oldest packet in the "waiting for an ACK" queue has exceeded the fine grained timer and if so retransmit the packet (therefore, Vegas records the time that each packet is sent into the network):
    - When the first or second duplicate ACK is received.
    - When the first or second nonduplicate ACK following a retransmission is received.

The improvements gained by these changes are greatest when multiple packets are lost. Brackmo and Peterson claim that these changes can reduce the number of time-outs by an additional 50% as compared to TCP/Reno.

The more significant contribution offered by Vegas is a congestion avoidance algorithm known as CAM. Based loosely on Tri-S, CAM monitors changes in throughput and incorporates the information in the slow-start and congestion avoidance algorithms. The design of Vegas reflects a fundamental requirement of any proposed change to TCP: the behavior of slow-start and congestion avoidance must be preserved, however, the algorithms can be enhanced in such a way to reduce send rate oscillations and to avoid packet loss. The challenge is to ensure that improvements gained by an enhancement do not come at the expense of other TCP (i.e., other than TCP/Vegas) connections.

Tri-S (while in increase mode) decides to decrease the rate (by reducing the current window by 1 packet) when the measured throughput is ½ the value sampled from the previous round trip time. Until this point, Tri-S will increase the window by 1 segment every round trip time. Vegas is similar in that it monitors throughput during congestion avoidance. However it uses throughput differently in the rate adjustment algorithm. Instead of looking at changes in throughput, Vegas compares the measured throughput with an expected throughput.

The algorithm defines the following variables:

- *BaseRTT* : The round trip time (RTT) of the connection when it is not congested. Essentially, the connection sets this variable to be the minimum round trip time observed by the connection throughout its lifetime.
- *Expected_Throughput = Current_Window_Size / BaseRTT*. Assuming the connection is not exceeding the available bandwidth, the *Current_Window_Size* is the number of bytes sent during

the minimum RTT. The *Expected_Throughput* is simply the amount of data sent during an interval of time (i.e., the time interval is the *BaseRTT*).

- *Actual_Throughput* : This parameter is updated each round trip time by recording the sending time for a segment being timed, recording the number of bytes transmitted between the time the segment is sent and when its ACK is received, and dividing the number of bytes transmitted by the sample RTT. If the network is uncongested, the *Actual_Throughput* equals the *Expected_Throughput*. As congestion builds, the measured RTT grows causing the *Actual_Throughput* to become less than the *Expected_Throughput*.
- *Diff=Expected_Throughput - Actual_Throughput :* This variable specifies the change in throughput that occurred during the last round trip time.

Vegas defines two threshold values for the *Diff* variable (α and β). The second change offered by Vegas modifies the congestion avoidance algorithm as follows:

Every other round trip time, measure the *Diff*
If *Diff* < α then increase *cwnd* linearly during the next round trip time
If $\alpha \leq$ *Diff* $\leq \beta$ then do nothing
If *Diff* $\geq \beta$ then decrease the congestion window linearly during the next round trip time

The third change offered by Vegas reduces the slow-start rate of increase by a factor of 2 and utilizes CAM to prevent packet loss. Specifically, Vegas modifies the slow-start algorithm as follows:

Every other round trip time, enable exponential growth
Measure the *Diff* during after a round trip time without growth
If *Diff* $\leq \beta$ then transition to congestion avoidance.

The performance of TCP/Vegas as observed in [12] is impressive. Vegas yields higher network throughput and transfers bytes more efficiently than TCP/Reno. Ahn et. al., confirms most of these results, however they did observe that in head-to-head transfers with TCP/Reno, Reno steals bandwidth from Vegas [1]. Intuitively this makes sense since Vegas is simply not as aggressive as TCP/Reno.

The Internet community has received TCP/Vegas somewhat apprehensively. Concerns associated with Vegas include:

- In the worst case, Vegas tracks congestion caused only by itself rather than from other sources. For example, if a Vegas connection begins over a path where a router has sustained congestion, Vegas would not detect congestion. Floyd stresses that in the absence of prior information about the fixed propagation delay of a path, it is not possible for end nodes to distinguish between propagation delay and persistent queueing [25]. As Internet router queues increase their size to handle more traffic and larger pipes, it becomes very important to not allow persistent queues to develop. Floyd states that the only way to accomplish this is through gateway congestion control.
- It is unclear how well CAM scales at higher speeds. Further study is required to see if CAM can track queue levels accurately for different bottleneck link speeds. For example, at higher speeds, the parameters (i.e., α, β and *Diff*) become small. It seems that the algorithm could be susceptible to noise due to potentially large fluctuations in the measured round trip times (perhaps caused by CPU scheduling delays at the endpoints).
- When a path switch occurs, the *Base_RTT* is no longer valid. Furthermore, if the new path is longer than the previous path, the *Base_RTT* will remain incorrect for the lifetime of the connection (or until the next path switch).
- The Vegas assessment of congestion (based on measured throughput) also includes congestion that occurs in the reverse path (experienced by the ACK packet). Vegas might unnecessarily react to congestion that occurs the ACKs flowing in the reverse path. The algorithm will not work

properly over asymetric paths where the forward and reverse paths differ significantly in bandwidth. Furthermore, the round trip time measurement (which is fundamental to the throughput calculation) includes delays experienced at the receiver (caused by either operating system scheduling latency or by delayed ACKs). If there is a sudden increase in receiver processing delays, Vegas will assume network congestion.

- Vegas can not guarantee fairness over connections with different path lengths since the throughput measurements are based on current window size which is proportional to the round trip time (as in base TCP). Furthermore, when multiple Vegas connections compete for bandwidth, it seems possible that they might all enter the "do nothing" state during congestion avoidance. If this occurs, each connection will continue to use the bandwidth it had upon entering the state not allowing the system to converge such that each connection has a fair share.

2.4 End-to-End Rate-Based Congestion Control

In the late 1980's researchers addressed the question of how to best support gigabit networking. One group went in the direction of inventing "lightweight" protocols optimized for gigabit speeds [80]. The underlying assumption was that existing transport protocols such as TCP were not capable of achieving gigabit speeds justifying the need for a new protocol. The majority of the proposed transport protocols reflected a common belief that window-based control is inferior to rate control at high speeds (e.g., VMTP [17], NETBLT[19] and XTP[79] all chose rate control).[16] The reasoning behind this belief is summarized as follows:[17]

- Window schemes can not guarantee a minimum rate (since they are dependent on the behavior of the network and the receiver to provide credits). Therefore, windows are inadequate for stream oriented, real-time applications (e.g., audio and video applications) that require minimum guaranteed rates. Real-time applications need guarantees based on a rate rather than based on a "count" provided by window control.
- Window schemes are prone to potentially bursty behavior (again, since they are dependent on the behavior of the network and the receiver). This leads to unstable network behavior resulting in highly variable response times.
- Windows schemes are inherently unfair to long path connections (compared to short path connections) since the throughput of a connection with a given window size is proportional to its round trip time [6]. However, two rate-based connections that have the same rate will have the same throughput regardless of round trip times.[18]
- While more of an observation than a reflection of a problem associated with window control, Jain states that the movement towards rate control is justified since the bottleneck at a router is no longer buffers but rather links and processing speeds [42]. The latter resources are rate limited in

[16]In fact each scheme utilizes dynamic rate control where the rate changes are based on either explicit or implicit feedback.

[17]The traditional problem pointed out with window control is that over large bandwidth-delay paths, when congestion occurs, the feedback arrives at the sender too late to avoid (potentially) significant packet loss [48]. However, this concern really is a statement that open-loop control with bandwidth reservation is required for gigabit speeds rather than closed-loop control over a connectionless network. The discussion in this section focuses on the trade-offs between window and rate control in a "best effort" network.

[18]The point here applies to an open-loop, static, window or rate-based scheme. In a closed-loop, dynamic scheme, if the rate (through either a window or a rate-control mechanism) is adjusted each time feedback arrives at the sender, short distance connections will increase their send rate quicker than a longer path connection. Other aspects of the congestion control algorithm determine if the scheme exhibits bias against longer path connections.

the sense that they can not sustain bits or packets arriving at a rate faster than their capacity. Memory used to be the bottleneck and it was count limited in the sense that it could not sustain more than a certain number of packets regardless of how fast or slow they arrive. Window control originated from the desire to keep the bottleneck memory from overflowing (as this was the limited resource). Modern routers come equipped with a relatively large amount of memory allowing for higher bandwidth utilization (resulting from rate control) at the expense of occasionally large queue levels.

The arguments against rate control include implementation difficulties (i.e., dependent on the system timer granularity) and the fact that rate control does not limit the maximum number of packets outstanding in the network by a source. A set of researchers who shared this bias against rate control took a different approach at high speed transport protocols by focusing on refinements to TCP and proving that it can work at high speeds. Improvements such as header prediction and enhanced (i.e., quicker) checksum calculations proved fruitful as several studies were able to demonstrate that TCP could approach gigabit speeds.[19] After this work, the interest in "lightweight" transport protocols diminished with the exception of IBM's Rapid Transport Protocol which we discuss shortly [55].

As stated earlier, rate control is an attribute of a congestion control scheme. It can be used in either open-loop or closed-loop schemes. In its purest form, rate-based congestion control implies open-loop control in a connection-oriented network. In such a scheme, an explicit rate is negotiated between the sender, the network and the receiver. Jain correctly points out that this type of scheme is actually a hop-by-hop level of congestion control since all routers along the path are aware of the parameters and must enforce them [42].

As noted earlier, an open-loop, connection-oriented scheme can underutilize network resources if the source traffic is not accurately characterized. Given our interest in the Internet, we will focus on closed-loop, rate-based congestion control schemes for connectionless networks. In particular, we want to explore the tradeoffs associated with rate and window control in the context of end-to-end transport protocols. We will conclude this section by exploring the potential benefits obtained by adding rate control to the current TCP protocol.

Closed-loop, rate-based congestion control consists of the following components:

- A traffic shaping mechanism is required to enforce the negotiated rate parameters (i.e., average send rate, maximum burst rate and maximum burst size). Note that in a connectionless, best-effort network, there will not be a policing function.
- A closed-loop control algorithm that includes the following:
  - A feedback scheme based either on explicit feedback from the network or implicit feedback deduced by the endpoints based on changes in traffic conditions.
  - A rate adjustment algorithm.
- A rate-control scheme should be reinforced with an end-to-end credit scheme that limits the sender's outstanding packet count (as recommended by Schwartz [51]).

Several different traffic shaping mechanisms have been proposed [64]. The attributes of the flow specification required by the network determines which mechanism is used (i.e., some networks might only specify a minimum gap in between packets while other networks might specify both a maximum rate and a maximum burst size). Leaky bucket is a simple scheme designed to meter data into a network at a controlled rate. The *(r,T)-Smooth* traffic model allows the connection to inject *r* bits of data into the network each *T* bit times. A token bucket mechanism is based on leaky bucket, however it tolerates a certain amount of burstiness by allowing credits to accumulate (to a maximum level) permitting the source to inject a bounded level of burstiness into the network. A token bucket can be enhanced by adding

[19]For example, Partridge notes that Cray's standard TCP/IP implementation has been measured sending data at 790Mbps [64].

a leaky bucket rate control stage after the token bucket mechanism to protect the network from large bursts. The additional control stage meters packets into the network at some maximum rate.

The following summarizes a number of closed-loop, rate-based schemes that have been proposed over the past decade. The most common traffic shaping scheme is a token bucket where the end node or the network defines the maximum burst size. Several of the schemes rely on a window scheme to bound the maximum number of packets outstanding in the network. Each algorithm assumes a connectionless network with the exception of Packet Pair and XTP. The following briefly summarizes each scheme's congestion control strategy:

- The Packet Pair algorithm assumes each router in the network implements a fair queueing algorithm [45]. Therefore, the assumption is that future networks will be connection-oriented. Assuming the network allocates each connection a service rate (which changes as connections come and go), packet pair is designed to adapt the source rate to an estimate of its allocated service rate. The source sends out two packets back-to-back. The ACKs should come back separated by the time it takes the second source packet to be serviced by the bottleneck link (i.e., the transmission time plus processing delay). The algorithm adds a window scheme in addition to the rate probing algorithm. The maximum window is set to the bandwidth * delay product where the bandwidth is the sampled service rate and the delay is the round trip time measurement.

- The Q-Bit scheme extends DECbit with a rate control algorithm [72] and is therefore based on explicit gateway congestion control. A benefit of Q-Bit (over DECbit) is that the scheme is fair even among different path length connections.

- The Adaptive Admission Congestion Control (AACC) scheme adjusts the rate by observing changes in the measured delay of probe packets [33]. It is therefore an end-to-end scheme that does not require assistance from the network. The algorithm is unique in that it measures the congestion that exists in a one-way path. The sender timestamps each probe packet. When the probe packet arrives at the receiver, it calculates a *virtual_delay* (the receiver's current time minus the timestamp from the sender). The receiver attaches the *virtual_delay* on to a reply packet to the original probe packet. When the sender receives the reply packet, it compares the *virtual_delay* of the current reply packet with that of the previous reply packet. If the change between the two *virtual_delay* values is positive, this is an indication that congestion has occurred over the one-way path between the sender and the receiver during the last round trip time. The algorithm filters noise caused by spurious events and makes a binary decision that either the network is congested or not. The sender uses this information in its rate adaptation algorithm.

- XTP was designed for high performance and with ease of VLSI implementation as a primary objective [79]. XTP offers a connection oriented, reliable transport data service using a selective repeat end-to-end error recovery procedure. XTP uses a dynamic, rate-based mechanism where the sender adjusts its send rate based on explicit rate feedback from either the receiver or from the network. XTP control packets contain a field for the send rate and burst size. Any node within an XTP path can modify the rate contained in the packets. Therefore, before queue levels become excessive, XTP routers can instruct an XTP sender to reduce its rate.

- IBM's Rapid Transport Protocol is a "lightweight" transport protocol designed for use in a High Performance Network (i.e., an SNA/HPR network) [12]. The protocol uses an adaptive rate-based algorithm (known as ARB) that adapts the send rate based on current traffic conditions. The ARB sender adds to a periodic probe packet the time difference since the time that the previous probe packet was transmitted (this is referred to as a measurement interval). When a probe packet arrives at the receiver it calculates its observed measurement interval (i.e., the time since the last probe packet arrived) and compares to the measurement interval sent by the sender. If the receiver's measurement interval is larger than the sender's, this indicates the probe packet experienced an increase in congestion compared to the congestion level of the previous probe

packet. ARB is a distributed algorithm that allows the ARB receiver to periodically sample the network congestion level (the sample frequency is about once each round trip time) in the path between the sender and the receiver.

We conclude this section by exploring the potential benefits of a modified version of TCP that supports rate-based congestion control. The idea of a reliable, rate-based transport protocol for the Internet was first explored by the NETBLT protocol [19]. One of the goals of NETBLT was to determine the feasibility of a rate-based transport protocol. The conclusion was that accurate rate control is possible although performance is constrained by the system timer granularity. More specifically, the traffic shaping requirements imposed by the network plus the system timer granularity essentially determines the maximum rate supported by the connection.

Recently, there has been renewed interest in applying rate control to TCP. Several researchers have observed that packet loss can be reduced if the sender's response to 'big ACKs' (a big ACK is an acknowledgment packet that acknowledges more than 2 segments) is controlled. For example Floyd implemented a '*maxburst*' parameter that limits to four the number of packets that can be sent in response to a single incoming ACK [21]. This enhancement is actually an example of burst control rather than rate control.

In addition to the TCP/Vegas CAM scheme, Brackmo and Peterson also proposed a rate probing scheme during slow-start to avoid packet loss (CAM might not be effective during slow-start, especially during the initial slow-start when the *ssthresh* is set very high) [12]. Their idea is to use the packet pair rate probe mechanism, however instead of using 2 sequential packets to generate a sample of bottleneck capacity, they use 4 sequential packets per sample. The time between the returned ACKs is measured to estimate the available bandwidth. Vegas uses this as an upper limit on its send rate. Based on limited simulations, Brackmo and Peterson found no significant improvement gained by adding rate probing to their congestion avoidance mechanism during slow-start.

The Vegas CAM algorithm is essentially a combined rate-based and window scheme. However, given that Vegas relies on ACKs to meter packets into the network, the scheme is much more of a window scheme than a rate-based scheme. A true rate control scheme by definition must either replace or augment TCP's ACK metering with some rate control mechanism (i.e., token bucket). The "rate" could be based on the *Expected_Throughput* as defined by Vegas. An interesting spin of this is a rate-based version of TCP/Vegas. The benefits include controlling the level of burstiness by a source and reducing global synchronization effects. The usefulness of a rate-based version of TCP is an open issue whose value hinges on how much burstiness truly exits in IP networks.

We conclude this section with a brief description of the advantages of an enhanced UDP protocol that supports an end-to-end, rate-based congestion control mechanism. A significant problem associated with TCP/IP is that a UDP application (or any flow that ignores congestion indications) can flood a network, starving TCP connections. Floyd has proposed link-sharing router services as a way to protect the Internet from the growing volume of MBone traffic [27]. More recently, Floyd has enhanced the RED algorithm to try to identify misbehaving flows and limit their throughput without requiring the router to keep per connection state [26]. A third solution recently proposed by Mahdavi and Floyd is for adaptive applications (which can monitor packet loss via some application level protocol) to adapt their send rates (i.e., dynamic rate control) based on theoretical models of TCP throughput as a function of segment size, round trip time and packet loss rates [52, 62]. The idea behind Mahdavi's scheme is to make sure that non-TCP applications react to congestion in a manner similar to TCP thereby assuring fairness. The first two proposals require changes to the routers and the Mahdavi suggestion requires cooperative applications. Protection from greedy sources at the router has advantages. Likewise the intuition behind Mahdavi's suggestion is good.

Another solution to this problem is to add a congestion avoidance scheme to UDP (or to a higher level unreliable transport protocol such as the Real-Time Protocol). Given that real-time applications are more amenable to rate control rather than to window control, a logical choice for a preventive congestion control mechanism for UDP is a rate-based congestion avoidance scheme. The key advantage is that such a scheme adds preventive congestion control at the transport layer protecting the network from high bandwidth (uncontrolled) applications. As long as the control scheme behaves in a similar manner as TCP, the UDP connection would compete fairly with TCP traffic. The transport layer can relay congestion indications back to the applications who might chose to adapt in the same way as current adaptive applications do. Additional study is required to understand the appropriate interaction between real-time adaptive applications and congestion feedback from the transport layer.

3. Integrated Services Architecture

Due to advances in technology, networked multimedia applications have exploded in popularity. The first set of real-time applications (i.e., vic and vat [20]) were intended to support audio and video streams over an IP network. These UDP applications depend solely on an IP network's best-effort service to provide a usable level of service. Vat is unique in that it is an "adaptive" application that adjusts the playback offset delay (i.e., the length of time that data is collected by the receiver before it is played back) to dynamically improve audio quality based on observed delay changes. However, the usability of even the latest set of sophisticated "adaptive" applications such as Real Audio [68] and Bamba [37] are highly dependent on network behavior.

Recognizing this, the Internet community formed a set of Integrated Services Task Forces responsible for defining enhancements that would allow IP networks to support real-time applications. The vision of an Integrated Services Architecture (ISA) was formally introduced in RFC 1663 in [8].

The underlying mission of the ISA effort is to define the next generation traffic management scheme for IP networks. At the highest level, the functions provided by an Integrated Services Network include:

- Real-time quality of service (QoS).
- The ability to divide traffic into classes and control the sharing of bandwidth over particular links (i.e., controlled link sharing).

In this section we provide a summary of the proposed ISA. We overview current approaches at realizing several of the key components and then we focus on the congestion control implications. In particular we want to focus on the proposed service enhancements (i.e., controlled-load and guaranteed service) and the resulting end-to-end requirements at the transport layer. Finally, we summarize the initial ISA deployment and project possible usage scenarios.

3.1 Summary of the Proposed Architecture

Braden, et. al., define the term Integrated Services (IS) as an Internet Service model that includes best-effort service, real-time service and controlled link sharing [8]. Essentially, this is a very concise requirements statement for an abstract IS model and in fact reflects the functions required of an Integrated Services Network that we identified earlier. The requirements imply that different levels of service are required in order to support different types of applications. For example, a real-time audio playback application has significantly different network requirements than does an ftp application. The

[20]Both vic and vat can be found at ftp://ftp.ee.lbl.gov/conferencing.

requirements also reflect the need to be able to manage particular flows. For example, one organization might want 50% of the bandwidth of a particular link and the remaining bandwidth might be shared equally by other users.

In order for the network to provide service guarantees or to implement link sharing, a fundamental Internet philosophy shift needs to occur: routers must reserve resources. This implies that:

- A flow setup mechanism is required.
- Routers must maintain state (as we will see, this might not require state per each connection).

The Integrated Services Architecture (ISA) specifies both an IS model as well as a reference implementation framework that provides at least a vocabulary and high level functional organization that realizes the IS model. The goal of the framework is to allow discussion of implementation issues without mandating a single design.

In the following discussion of the IS model and framework, we first want to define the terms 'flow' and 'session'. A flow is a distinguishable stream (i.e., a connection) and a session is a group of related connections (e.g., a multicast group). A flow is created in an IS network when an application requests a connection with a partner using a specified quality of service. The QoS is specified by a list of parameters called a flowspec (formerly referred to as a Tspec). A Tspec includes the token bucket parameters (bucket depth and bucket rate) in addition to a peak rate, a minimum policed unit and a maximum datagram size.

The service aspects of the IS model (i.e., the services required of the model) include the following:

- QoS requirements: A taxonomy of network applications reveals three fundamental categories of applications each with a unique service requirement. Elastic applications will tolerate packet delays and are satisfied with a 'best effort' service. Intolerant applications require absolute maximum delay bounds and therefore require a guaranteed service. Tolerant applications will tolerate some delay and are satisfied with a predictive service which supplies a fairly reliable delay bound.
- Resource sharing: An IS router must support the three QoS service levels as specified above. The IS router must also support link sharing. The router must share link bandwidth fairly over all flows taking into account different classes.
- Packet dropping: The IS network might chose to mark packets from some flows as preemptable whereby some of the packets are discarded when the network reaches the point where it can not meet all of its service commitments.
- Usage monitoring: The IS network must monitor source traffic and be able to prevent one single flow from abusing network resources. Furthermore, the network might provide some feedback to users to providing a mechanism for a flow to voluntarily adapt its behavior.
- Reservation model: This describes how an application negotiates for a QoS level.

The following summarizes the components of an ISA framework that have been proposed to realize the required services of the IS model:

- Packet scheduling (PS): The PS manages forwarding different packet streams using either class-based or priority queueing (or more likely, a hierarchical approach using both of these two techniques). The PS component implements the QoS as well as the controlled link sharing functions.
- Bandwidth Enforcement: The flowspec of each flow must be enforced. The function can be implemented either at the source, at a network access node or within the network.

- Classifier : Incoming packets must be mapped into a class where all packets of the same class get equal treatment by the PS. The classifier is the component of the framework that maps certain fields of a packet (e.g., destination address/multicast group address and protocol ID) to an established QoS class that is used by the PS component.
- Admission control: The admission control service of the IS model is the mechanism that decides if a network can support a flow request with a given flowspec. Routers must monitor current usage levels (either statically based on all flowspecs of active flows or dynamically by monitoring flow dynamics). A router might decide that it will not support the requested flow or it might agree to the flow but only with a modified flowspec (e.g., reduced send rate).
- Reservation setup protocol (RSVP) : RSVP is a resource reservation protocol providing the mechanism by which a one-way flow establishes the state at each router along the path required for the resource reservation [90,9].

The two elements that we will focus on for the remainder of this section are RSVP and the packet scheduling support in the router. In RSVP, A sender submits a request specifying its traffic characteristics in a flowspec. The request is examined by the destination (i.e., the receiver) and by all intermediate nodes and is either rejected, accepted or accepted with a modified flowspec. The main highlights of RSVP are summarized as follows:

- RSVP was designed for either unicast or multicast sessions. For a multicast applications, senders dynamically join and exit the session. The resource allocation along the path dynamically adjusts to changes in flows.
- A sender joins an RSVP session by sending a PATH message to the multicast group (or to the receiver if it is a unicast session) that contains the flowspec. The PATH message flows through the path to each receiver in the group. Routers along the path record the input link and output link(s) associated with the message. A receiver will receive each PATH message (from all flows) and forwards a RESERVATION message back to the sender that contains the aggregate resources required by each flow. Each router in the path receives the RESERVATION message and (based on an admission control algorithm) decides if the router can support the flow. If not, it denies the reservation request. If the router is able to support the flow, it directs the RESERVATION message back to the next router in the upstream path (by simply reversing the addresses of the state associated with the original PATH message). When the RESERVATION message arrives at the sender, it begins to utilize the “pipe”.
- Each router in the path maintains a “soft state” such that if the router loses state it will be automatically reinstated by RSVP. There are two types of soft state: path state and reservation state. Periodically, a source sends path messages that establish or updates the path state. Similarly, periodically the receiver sends RESERVATION messages to establish and refresh the reservation state associated with the session. A time-out is associated with the soft state such that if a flow disappears (without specifically exiting the session), a time-out occurs at each router in the path causing the resources associated with the flow to be released. This capability facilitates flows joining and leaving the session as well as handles synchronizing a new path (perhaps due to a link failure) with the required resource allocation.
- RSVP allows a receiver to specify which of the senders in a session of interest. For example, a receiver in a multisource video application can select which source(s) it wants to “tune in”. A receiver specifies particular senders by adding a packet filter to the RESERVATION request.

As previously mentioned, the packet scheduler in an Integrated Services router must support different levels of service as well as provide a link sharing function. We now outline a scheme that a router can use to implement these functions. It has been shown that a weighted fair queueing (WFQ [20]) scheduling discipline at a router accompanied by a token bucket filter at the source will provide an absolute upper

bound on the network delay of the source traffic [63]. The same weighted fair queueing scheme can also provide controlled link sharing of the available link bandwidth.

The router can achieve predictive real-time service by mixing same class traffic at the same queue. Therefore, two levels of traffic scheduling are required. One which separates traffic with different service objectives (best effort, guaranteed service and predictive service) and a second level that schedules each first level class using an algorithm to meet specific service objectives. One approach identified in [8] uses WFQ as the first level of policy and then preventive and best effort service levels are assigned priorities and processed accordingly. Flows in a predictive session can be further divided perhaps based on different levels of delay bound simply by adding additional priorities.

3.2 Congestion Control in ISA: Current Directions

The IETF has translated the QoS requirements defined by the IS model into two new service models: guaranteed service [86] and controlled-load [74]. The guaranteed service provides mathematically provable bounds on end-to-end datagram queueing delays. The service does not attempt to minimize jitter (i.e., the difference between the minimal and maximum delay of packets), instead it controls the maximum queueing delay. With the queueing delay bound and some knowledge of the latency of the path (e.g., service times), an upper bound on maximum packet delay can be computed. Clearly, the guaranteed service is conceptually similar to ATM's Constant Bit Rate (CBR) service. The service is for intolerant applications that will not accept datagrams that exceed a maximum delay bound.

Relating a guaranteed service to the congestion control framework presented earlier in the report, clearly the service is open-loop (and therefore preventive). A guaranteed service is similar to the classic open-loop control model where the network reserves enough resources for the application and relies on a rate-based server algorithms to maintain the rate with a bounded delay. Traffic shaping is required (either at the source or at the network access node) and a "reshaping" function at each router is also required to maintain a flow's Tspec parameters. A guaranteed service provides a network access level of congestion control although Jain correctly points out that a pure open-loop scheme (such as guaranteed service) is not only a network access level of control but also a hop-by-hop control scheme since all routers in the path must enforce each flow's rate [42]. The challenge of open-loop is to be able to accurately characterize source traffic. In a guaranteed service, if a source exceeds its traffic description, the excess traffic will be marked (by the traffic shaper) as nonconformant and consequently subject to the same jitter and packet loss as best-effort traffic in the network.

Given the costs associated with a guaranteed service (i.e., reserved bandwidth plus complex queue algorithms), and given the demonstrated ability of adaptive applications to not require a guaranteed rate, it is expected that the controlled-load service will attract more interest (at least for the initial deployment of ISA) than the guaranteed service. The IETF defines controlled-load service as one that approximates the behavior visible to applications receiving best-effort service under "unloaded conditions". Therefore, the applications can assume:

- A very low packet loss rate (that approximates the packet loss rate of the medium).
- A high percentage of packets will not experience queueing delay.

As with the guaranteed service, a controlled-load service relies on open-loop control such that a source must characterize its traffic (and reserve network resources using RSVP). However, controlled-load service is not a classic open-loop scheme as reshaping within the network is not performed. Therefore, unlike the guaranteed service, the controlled-load service only provides network access level control (without additional hop-by-hop control).

Sources using guaranteed and controlled-load services are allowed to exceed their reservation. Given the nature of applications requiring guaranteed service (i.e., real-time applications that require bounded delays), it would be abnormal for a guaranteed service application to purposely exceed its bandwidth reservation. On the other hand, it would not be uncommon for certain controlled-load applications to use as much bandwidth as possible. An issue not yet fully addressed is how to handle offending controlled-load applications with respect to other best-effort applications. For example, should controlled-load traffic that exceeds its traffic description be given higher or lower priority than other best-effort traffic? If the excess controlled-load traffic must compete fairly with best-effort traffic, this once again raises the issue of whether a rate-based, end-to-end congestion control scheme is required.

Currently, the majority of real-time audio/video applications use the Real-Time Protocol (RTP) [73] to transport real-time streams over IP using UDP. For the same reasons defined earlier when we proposed rate-based congestion avoidance for a UDP transport, it seems appropriate to investigate enhancements to RTP when used in a controlled-load service environment. For example, RTP can be extended to provide a rate control function that shapes traffic based on the requested Tspec and that adjusts flow parameters dynamically based on observed congestion.

Given that a sender who wants to use the controlled-load service has to go through a resource reservation process (i.e., RSVP), it is unlikely that widely used TCP applications such as web browsers will make use of RSVP. It is more likely that certain "niche" applications will be able to make use of RSVP. For example, it seems feasible for an Internet Service Provider (ISP) to offer a service that transports SNA traffic through an RSVP controlled-load pipe. Using Data Link Switching (DLSw [5]), the aggregate SNA traffic can be encapsulated in a single TCP connection over the "pipe". Even though routers available today typically treat the DLSw SNA traffic with higher priority over other IP traffic, the advantage of RSVP lies in the process of setting up such connections. Currently, each router in the path must be manually configured to give specific TCP connections (identified by a unique port number) priority. Furthermore, if too many connections are allowed to share the allocated bandwidth, the benefits of priority disappear. RSVP essentially automates the process of establishing and allocating reserved bandwidth.

Another potential use for TCP connections over RSVP controlled-load pipes is for enhanced tunneling solutions. Currently, it is possible for a corporation to interconnect sections of its network by establishing secure and reliable data pipes (i.e., tunnels) through the Internet. With a controlled-load service, an ISP can provide more reliable tunnels with guaranteed throughputs.

The point is that TCP will be used over RSVP pipes. This (again) raises the issue of a rate-based TCP protocol. Several recent studies suggest that simply applying a token bucket mechanism to a TCP stream (either at the source host or at a network access box) is not optimal [50,22]. Based on simulation, both studies found that for various reasons, TCP/Reno is not able to utilize all of the reserved bandwidth provided by a controlled-load service. Kuo found that TCP/Reno could adequately utilize reserved bandwidth over a controlled-load RSVP pipe (implemented using a a buffered token bucket with weighted fair queueing at the routers) as long as the ACK's were routed over a path of similar capacity. If the return path for the connection has lower available bandwidth, the connection will underutilize the reserved bandwidth. The work by Feng, et. al., observes that the packet clustering behavior of TCP causes a significant number of token bucket credits to essentially be thrown away (i.e., unused credits implies the allocated bandwidth is not fully utilized).

Feng, et. al., went one step further and added rate control to TCP (the authors actually referred to their scheme as a timed send mechanism). They simulated a controlled-load environment using a token bucket filter that marks packets that exceed a flow's Tspec. The authors implement a modified version of the RED algorithm that divides traffic into one of two classes: conformant and nonconformant. During congestion, the nonconformant traffic will be dropped with a higher probability than the conformant traffic. The rate control scheme augments the TCP ACK-based metering of packets with timer-based triggering. Whenever a periodic timer expires (i.e., a burst timer), the sender checks if it has tokens (tokens are being supplied at the send rate specified in the flow spec). If there are tokens, additional

packets are sent as long as the receiver's advertised window is not exceeded (however, the congestion window is ignored). Feng found that it was also necessary to limit the maximum burst allowed when the sender receives a "big ACK" (i.e., an ACK that acknowledges more than 1 packets). As described earlier, setting a *maxburst* parameter to something low (Feng chose a value of 2 packets) eliminates the burstiness. The authors conclude that with these enhancements, TCP is able to operate efficiently in a controlled-load environment.

We propose one additional step: add an end-to-end, preventive congestion control scheme that adjusts the TCP send rate based on either explicit or implicit feedback. Further study needs to confirm this, however we assert that a rate-based version of TCP that has additional congestion avoidance algorithms would, in some situations, be advantageous over just a rate-based version of TCP when used in an RSVP environment. Our reasons for this assertion include:

- As mentioned, some controlled-load applications will purposely try to exceed their Tspec. The excess traffic competes with other excess traffic (and possibly with lower priority best-effort traffic) for remaining bandwidth. A closed-loop preventive control scheme will detect the congestion and reduce the TCP source rate (to the minimum allowed rate specified by the Tspec). Further study needs to confirm this, but the additional congestion avoidance scheme should allow controlled-load connections to more efficiently utilize available bandwidth that exceeds their original reservation.
- In an RSVP session, there will be periods when the controlled-load service temporarily stops (replaced by best effort service) due to route changes. A congestion avoidance scheme will help to minimize the throughput reduction that might occur during these transient periods.[21]
- It will take many years before all IP networks support guaranteed services. While one leg of a TCP connection might pass through an Integrated Services Network, another leg of the connection might travel over a subnetwork that does not support ISA. Further study needs to confirm this, but we assert that in a heterogeneous network it is still possible to make use effective use of reserved bandwidth. In addition to rate control, an end-to-end congestion avoidance mechanism can enhance the performance of TCP connections over heterogeneous networks.

3.3. Status of the Integrated Services Deployment

Cisco has released support for RSVP in their routers last year (using weighted fair queueing to provide a controlled-load service). Bay networks has announced plans to release RSVP in their routers in the first half of 1997. There is at least one ISP (BBN Corporation) who offers an RSVP service. At least one content provider (Worldwide Broadcasting Network, WBN) has teamed with BBN to provide multimedia content to financial institutions and other corporations.

RSVP is not without flaws. A significant concern of RSVP questions the scalability of the protocol. Given that each flow requires a reservation and that flows are currently identified by a combination of source and destination addresses (plus several other fields in the IP header), a router will only be able to support a finite number of flows. Ongoing work is addressing this by finding more manageable ways to aggregate flows. A second concern of RSVP is that the reservation protocol is capacity based and excludes policy. Most networks will want to assign bandwidth to certain groups or users and restrict or limit access by other users (and ignore what the application is actually requesting). Work is in progress to extend RSVP's reservation model to include policy based admission control [35].

[21]Note that if the IETF decides to pin RSVP routes, this is no longer an issue.

RSVP will likely not be widely deployed within corporate Intranets for at least the next several years while the technology goes through an evaluation phase. Potential uses for Integrated Services within an Intranet include PC based videophones and videoconferencing capabilities. It is expected that the services sector will be more aggressive than the corporate sector in deploying ISA capabilities. ISPs will likely offer the following services over the next several years:

- Internet phone service
- Videophone service
- Audio/visual messaging service
- Education/training services
- Transport network services (e.g., transport SNA traffic over an ISP's network or "premium" IP services)

The Internet phone might be the "killer" application that drives the deployment of the Integrated Services technology into the Internet. The IDC estimates that 60 million PC users will be making voice (phone) calls over the Internet by 1999. A survey by Infotest indicates that 25% of the worlds phone calls will travel over the Internet within 6 years.

4. Conclusions

In this paper, we have presented a framework for describing congestion control. Based on this, we surveyed congestion control as it relates to TCP/IP. Of special interest were end-to-end congestion avoidance schemes (both explicit and implicit feedback) that could be applied to TCP. Also of interest were the advantages and disadvantages of rate control as compared with window control mechanisms.

Prior work observes that rate control is difficult to implement and that during times of network congestion, rate control can potentially flood a connectionless network. Advances in technology are helping to solve the first issue (faster workstations allow for finer grained timer granularity). A combined rate and window scheme can offer the advantages of rate control while bounding the maximum number of packets outstanding in the network.

We have pointed out that the Internet community is working on a set of enhancements (i.e., TCP/SACK, RED and TCP/Vegas) to TCP that (either separately or collectively) can provide significant improvement to TCP/IP networks. It is unclear if the benefits gained from a rate-based version of TCP would be significant when compared to the improvements provided by the future TCP enhancements. However, prior work has shown that a rate-based version of TCP is essential for reliable transport in an Integrated Services environment. We have asserted that additional benefits might be possible when a rate-based TCP protocol is combined with an enhanced congestion avoidance scheme when used over RSVP pipes.

We conclude this report by summarizing issues that we feel are worthy of additional study:

- There is the need for an end-to-end congestion avoidance scheme that is based on implicit feedback in a connectionless network (i.e., schemes that monitor congestion by tracking changes in throughput or round trip delays). Additional analysis is required to verify if the TCP/Vegas CAM algorithm is adequate.
- Further study is needed to explore enhancements to the Real-Time Protocol (i.e., RTP) to provide rate-based, preventive congestion control (both in the current IP environment and in an Integrated Services environment).

- Further study is required to explore reliable transport protocols that provide rate-based, end-to-end, preventive congestion control for RSVP pipes.

5. References

1. J. Ahn, P. Danzig, Z. Liu, L. Yan, “Evaluation of TCP Vegas: Emulation and Experiment”, ACM SIGCOMM95.
2. V. Ahuja, “Routing and Flow Control in Systems Network Architecture”, IBM Systems Journal, Vol 2, 1979.
3. K. Bala, I. Cidon, K. Schraby, “Congestion Control for High Speed Packet Switched Networks”, INFOCOM 1990.
4. H. Balakrishnan, in a note to end2end-interest, Available at ftp://ftp.isi.edu/end2end.
5. A. Bartky, L. Wells, “Data Link Switching: Switch-to-Switch Protocol AIW DLSw RIG: DLSw Closed Pages, DLSw Standard Version 1.0”, RFC1795, April 1995.
6. D Bertsekas, G. Gallager, “Data Networks”, Prentice Hall, 1992.
7. F. Bonomi, K. Fendick, “The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service”, IEEE Network, March/April 1995.
8. R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC 1663, July 1994.
9. B. Braden, et. al., “Resource Reservation Protocol (RSVP) - Version 1 Functional Specification”, Internet Draft, July 1996, <draft-ietf-rsvp-spec-13.txt>.
10. R. Braden, V. Jacobson, “TCP Extensions for Long-delay Paths”, RFC 1072, October 1988.
11. R. Braden, J. Postel, “Requirements for Internet Gateways”, RFC 1009, June 1987.
12. L. Brakmo, S. O’Malley, L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance”, ACM SIGCOMM94, 1994.
13. L. Brakmo, L. Peterson, “Performance Problems in BSD4.4 TCP”, ACM Computer Communications Review, October 1995.
14. W. Brogan, Modern Control Theory, Prentice Hall, 1991.
15. CCITT 1992, “Recommendation Q.922, ISDN Data Link Layer Specification for Frame Mode Bearer Service”, CCITT, Geneva, 1992.
16. A. Charny, D. Clark, R. Jain, “Congestion Control with Explicit Rate Indication”, ICC, V3 1995.
17. D. Cheriton, C. Williamson, “VMTP as the transport layer for high-performance distributed systems”, IEEE Communications Magazine, V27, No.6, 1989.
18. D. Chiu, R. Jain, “Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks”, Computer Networks and ISDN Systems, 1989.
19. D. Clark, M. Lambert, L. Zhang, “NETBLT: A Bulk Data Transfer Protocol”, RFC 998, 1987.
20. A. Demers, S. Keshav, S. Shenker, “Analysis and Simulation of a Fair Queueing Algorithm”, Journal of Internetworking Research and Experience, October 1990.
21. K. Fall, S. Floyd, “Comparison of Tahoe, Reno and SACK TCP”, Computer Communication Review, July 1996.
22. W. Feng, D. Kandlur, D. Saha, K. Shin, “TCP Enhancements for an Integrated Services Internet”, IBM Research Report RC 20617.
23. S. Floyd, “Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-Way Traffic”, ACM Computer Communication Review, 21(5), October 1991.
24. S. Floyd, “Simulator Tests”, Technical report, Available at http://www.nrg.ee.lbl.gov/nrg-papers.html, July 1995.
25. S. Floyd, “TCP and Explicit Congestion Notification”, ACM Computer Communications Review, October 1994.
26. S. Floyd, K. Fall, “Router Mechanisms to Support End-to-End Congestion Control”, Draft available at ftp://ftp.ee.lbl.gov/papers/collapse.ps.

27. S. Floyd, V. Jacobson, “Link-sharing and Resource Management Models for Packet Switched Networks”, IEEE/ACM Transaction on Networking, Vol#3, #14, August 1995.
28. S. Floyd, V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking, August 1993.
29. S. Floyd, V. Jacobson, “Traffic Phase Effect in Packet-Switched Gateways”, ACM Computer Communication Review, 21(2), April 1991.
30. F. George, G. Young, “SNA Flow Control: Architecture and Implementation”, IBM Systems Journal, Vol 21, 1982.
31. M. Gerla, L. Kleinrock, “Flow Control: A Comparative Survey”, IEEE Transactions on Communications, Vol 28, No. 4, April 1980.
32. A., Giessler, J. Hanle, A. Konig, E. Pade, “Free Buffer Allocation - An Investigation by Simulation”, Computer Networks, Vol 1, No 3, July 1978.
33. Z. Haas, “Adaptive Admission Control”, ACM SIGCOMM91, 1991.
34. E. Hashem, “Analysis of Random Drop Gateway Congestion Control”, Report LCS TR-465, Laboratory for Computer Science, MIT, Cambridge, MA, 1989.
35. S. Herzog, “Policy Control for RSVP: Architectural Overview”, Internet Draft, <draft-ietf-rsvp-policy-arch-01.txt>.
36. J. Hoe, “Improving the Start-up Behavior of a Congestion Control Scheme for TCP”, ACM SIGCOMM96, 1996.
37. IBM Corporation-AlphaWorks: http://www.alphaworks.ibm.com/. Bamba from AlphaWorks, 1996.
38. V. Jacobson, “Congestion Avoidance and Control”, ACM SIGCOMM88, 1988.
39. V. Jacobson, R. Braden, D. Borman, “TCP Extensions for High Performance”, RFC 1323, May 1992.
40. R. Jain, “A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks”, Digital Equipment Corporation Technical Report DEC-TR-566, April, 1989.
41. R. Jain, “Congestion Control in Computer Networks: Issues and Trends”, IEEE Network Magazine, May 1990.
42. R. Jain, “Myths About Congestion Management in High-Speed Networks”, Internetworking: Research and Experience, 1992.
43. J. Jaffe, “Flow Control Power is Nondecentralizable”, IEEE Transactions on Communications, V COM-29, #9, September 1981.
44. P. Karn, C. Partridge, “Improving Round-Trip Time Estimates in Reliable Transport Protocol”, ACM SIGCOMM87, October, 1987.
45. S. Keshav, “A Control-theoretic Approach to Flow Control”, ACM SIGCOMM91, September 1991.
46. L. Kleinrock, “On Flow Control in Computer Networks”, Proceedings of the International Conference on Communications, Vol 2, June 1978.
47. L. Kleinrock, “Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications”, ICC 1979.
48. L. Kleinrock, “The Latency/Bandwidth Tradeoff in Gigabit Networks: Gigabit networks really are different!”, IEEE Communications Magazine, April 1992.
49. H. Kung, R. Morris, “Credit-Based Flow Control for ATM Networks”, IEEE Network Magazine, Vol 9, No 2., March/April 1995.
50. T. Kuo, “TCP Dynamics under the Integrated Service Internet”, Submitted to Globecom97.
51. T. La Porta, M. Schwartz, “Architectures, Features, and Implementation of High-Speed Transport Protocols”, IEEE Network Magazine, May 1991.

52. J. Mahdavi, S. Floyd, “TCP Friendly Unicast Rate-Based Flow Control”, Technical note sent to End2end interest group, January 8, 1997.
53. A. Mankin, “Random Drop Congestion Control”, ACM SIGCOMM90, 1990.
54. A. Mankin, K. Ramakrishnan, “Gateway Congestion Control Survey” , RFC 1254, 1991.
55. J. Martin, A. Nilsson, “Congestion Control in HPR”, Submitted to GLOBECOM97.
56. M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgment Options”, RFC 2018, October 1996.
57. R. McQuillan, I. Richer, E. Rosen, “The New Routing Algorithm for the ARPANET”, IEEE Transactions on Communications, Vol COM-28, No. 5, May 1980.
58. P. Mishra, H. Kanakia, “A Hop by Hop Rate-based Congestion Control Scheme”, ACM SIGCOMM’92, September 1992.
59. D. Mitra, J. Seery, “Dynamic Adaptive Windows for High Speed Data Networks”, ACM SIGCOMM90.
60. J. Mogul, “Observing TCP Dynamics in Real Networks”, ACM SIGCOMM92, 1992.
61. J. Nagle, “Congestion Control in IP/TCP Internetworks”, ACM Computer Communications Review, Vol 14, No 4, October, 1984.
62. T. Ott, J. Kemperman, M. Mathis, “Window Size Behavior in TCP/IP with Constant Loss Probability”, DIMACS Workshop on Performance of Realtime Applications on the Internet, Nov 1996.
63. A. Parekh, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks”, Techinical Report LIDS-TR-2089, Laboratory for Information and Decision Systems, MIT, 1992.
64. C. Partridge, “Gigabit Networking”, Addison-Wesley, 1994.
65. H. Perros, K. Elsayed, “Call Admission Control Schemes: A Review”, IEEE Communications Magazine, November 1996.
66. J. Postel, “Transmission Control Protocol”, RFC 793, 1981.
67. J. Postel, “Internet Control Message Protocol”, RFC-792, September 1981.
68. Progressive Networks: http://www.realaudio.com/. Progressive Networks, the Home of RealAudio, 1996.
69. K. Ramakrishnan, D. Chiu, R. Jain, “Congestion Avoidance in Computer Networks with a Connectionless Network Layer. Part IV: A Selective Binary Feedback Scheme for General Topologies”, Digital Equipment Corporation, Technical Report TR-510, August 1987.
70. K. Ramakrishnan, R. Jain, “A Binary Feedback Scheme for Congestion Avoidance in Computer Networks”, ACM Transactions on Computer Systems, vol8, May 1990.
71. C. Rohrs, R. Berry, S. O’Halek, “A Control Engineer’s Look at ATM Congestion Avoidance”, GLOBECOM95.
72. O. Rose, “The Q-bit Scheme”, ACM Computer Communications Review, April 1992.
73. H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, Internet Draft, March 1995, <draft-ietf-avt-svc-rtp-07.txt>
74. S. Shenker, C. Partridge, R. Guerin, “Specification of Guaranteed Quality of Service”, Internet Draft, Febuary 1997, <draft-ietf-intserv-guaranteed-svc-07.txt>.
75. S. Shenker, L. Zhang, D. Clark, “Some Observations on the Dynamics of a Congestion Control Algorithm, “ACM Computer Communication Review, 20(4), October 1990.
76. W. Stallings, “ISDN and Broadband ISDN”, Macmillan Publishing Company, 1992.
77. W. Stevens, “TCP/IP Illustrated, Volume 1 The Protocols”, Addison-Wesley, 1994.
78. W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”, RFC 2002, January 1997.

79. W. Strayer, B. Dempsey, A. Weaver, “XTP: The Xpress Transfter Protocol”, Addison-Wesley, 1992.
80. A. Tantawy, “High Performance Networks: Technology and Protocols”, Kluwer Academic Publishers, 1990.
81. J. Turner, “New Directions in Communications (or Which Way to the Information Age?)”, IEEE Communications Magazine, 24, October 1986.
82. C. Villamizar, C. Song, “High Performance TCP in ANSNET”, ACM Computer Communications Review, October 1994.
83. Z. Wang, J. Crowcroft, “A New Congestion Control Scheme: Slow-start and Search (Tri-S), ACM Computer Communication Review, V21 #1, January 1991.
84. Z. Wang, J. Crowcroft, “Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm”, ACM Computer Communication Review, April 1992.
85. R. Wilder, K. Ramakrishnan, A. Mankin, “Dynamics of a Congestion Control and Avoidance of Two-way Traffic in OSI Testbed”, ACM Computer Communication Review, 21(2), April 1991.
86. J. Wroclawski, “Specification of the Controlled-Load Quality of Service”, Internet Draft, November 1996, <draft-ietf-intserv-ctrl-load-svc-04.txt>.
87. J. Wroclawski, “The Use of RSVP with IETF Integrated Services”, Internet Draft, October 1996, <draft-ietf-intserv-rsvp-use-01.txt>.
88. C. Yang, A. Reddy, “A Taxonomy for Congestion Control Algorithms in Packet Switching Networks”, IEEE Network, July/August 1995.
89. L. Zhang, “VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks” Proc SIGCOMM ‘90, September, 1990.
90. L. Zhang, S. Deering, D. Estring, S. Shenker, D. Zappala, “RSVP: A New Resource ReSerVation Protocol”, IEEE Network, September 1993.
91. L.Zhang, S.Shenker, D. Clark, “Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-way Traffic”, ACM SIGCOMM91, 1991.