Congestion Control in Frame Relay Networks
Using Explicit Binary Feedback

Fred R. Goldstein

Digital Equipment Corporation
550 King St. LKG2-2/N11
Littleton MA 01460 USA

Abstract: While Frame Relay networks are nominally connection-oriented, they lack the explicit flow control mechanisms usually associated with virtual circuits. This can lead to potentially serious congestion problems if users do not moderate their traffic. Individual users may present a wide range of traffic demand, which in practice is often unpredictable.

Explicit Binary Feedback is a congestion management technique that operates over a wide range of operating conditions in connectionless networks as well as across Frame Relay networks. It relies upon a one-bit indicator in each frame which probes the congestion state of all network resources that it encounters. If any node is congested, then the end-to-end virtual circuit user reduces its demand; if no congestion is seen, the user may increase demand. This adjustment can be performed in the Data Link layer elements of procedure or in the Transport layer, using window size as a surrogate for offered data rate.

1. Introduction

While Frame Relay is a relatively new service, very little of what goes into it is really new at all. Frame Relay is more of a novel combination of elements found in several older network protocols and services. In developing techniques for making use of Frame Relay in peer-to-peer network environments such as DECnet/OSI and TCP/IP, Digital has been able to make use of many existing technologies. This allowed us to develop a flexible, stable and simple congestion management technique for sending data over Frame Relay networks.

Of particular interest is the way in which Frame Relay combines aspects of both connection-oriented and connectionless networks. Frame Relay is nominally connection-oriented, because it makes use of a virtual channel identifier that is negotiated before data transfer can begin. That virtual channel may also have a negotiated throughput class. Both of these features are of course found in traditional packet-switched networks, such as those described in CCITT Recommendation X.25.

But unlike X.25, Frame Relay has no explicit protocol mechanism for flow control. While X.25 provides both sliding window and stop-go flow control in both its Level 2 and Level 3 protocols, Frame Relay has neither. This may simplify the implementation of the switching system, but leaves open the question of how users are to know whether or not the network is ready, willing and able to accept additional frames.

This problem is of course not unique to Frame Relay. Indeed, connectionless networks by definition do not provide explicit flow control! They may provide implicit means, such as admission controls and hop-by-hop (layer 2) flow control, but nothing equivalent to X.25’s Level 3. So in this regard, a Frame Relay network has more in common with a connectionless service than with a typical connection-oriented network.

From this follows the observation that a congestion control procedure that is effective for connectionless networks is likely to be effective in a Frame Relay environment. To be sure, a true connectionless network is nominally memoryless, with no throughput class enforcement. Frame Relay networks have the right to discard packets when congestion occurs, and can base that upon the negotiated parameters of a connection. But such implicit congestion notification by discard occurs by default anyway, just as it occurs in connectionless networks: If a network element runs out of buffer resources and has no way to provide flow control signaling (backpressure) to the sources, what alternative does it have but to discard data?

1.1. Nature of traffic on connectionless networks

Frame Relay service is quite versatile, allowing it to be applied to several very different applications. One obvious use for a Frame Relay network is to connect computer display terminals to distant host computers. This has been one of the oldest applications of computer networks and remains a favorite use of today’s packet-
switched networks. From a traffic management perspective this is actually one of the simplest problems to solve. Most terminals generate a stable, predictable level of traffic. An individual terminal's demand may vary widely over time, but on average, a group of terminals will generate a reasonably steady and predictable demand.

Another simplification that can be applied to some terminal-oriented networks is the use of a single end user terminal per network interface. If a Frame Relay terminal adapter connects a synchronous or asynchronous terminal directly to a network, then it can use local flow control procedures, as necessary, to enforce its throughput class.

These techniques cannot generally be applied to the sorts of traffic found on today's peer-to-peer connectionless networks. In most cases, the Frame Relay network is not connected to the actual source of the traffic (i.e., the mainframe computer, server or workstation generating the packets). Instead, Frame Relay provides a subnetwork service, connected to users via some sort of interworking unit, such as a bridge, router, or gateway. The connectionless network does not provide direct flow control. It must, however, have some mechanism for congestion control, lest it become unstable in times of high demand. That mechanism generally relies upon the cooperation of the users and the network.

Traffic on such host-based data networks is essentially unpredictable. The interworking unit has no idea of the number or nature of applications which might actually be establishing higher-layer connections. It simply forwards packets as it receives them, or discards them when heavily congested.

Another aspect of Transport layer protocols such as TCP and OSI Transport is that they are designed to seek out and optimize the maximum rate at which they can operate at any given time. They do not support rate-based flow control! Instead, they make use of a congestion control strategy in which the user gradually increases the offered load until one of several events stops it:

- Its physical access may become saturated,
- The sending or receiving node is incapable of running faster, or
- It detects congestion within the network.

No timers or flow metering are used; instead, the transport protocol and its supporting lower layers work in concert with the network to provide in effect a servo mechanism, constantly seeking out the ideal transmission rate.

This allows us to provide a rather simple approach to solving congestion control on Frame Relay networks. We basically extend the congestion management strategy that we use on connectionless network to cover the Frame Relay subnetwork. Ideally, the Frame Relay nodes use the same congestion avoidance procedures as the routers themselves do. We call this technique explicit binary feedback, and have incorporated it into DECSnet/OSI. It has also been adopted by the National Institute of Standards and Technology to be used with the OSI connectionless network service, working in conjunction with OSI Transport Class 4.

2. The first step: Implicit congestion control

Before explicit binary feedback was developed, networks needed a congestion control mechanism that was capable of operating over a wide range of speeds. We discovered the importance of this when Ethernet local area networks were becoming popular. Before then, most network connections, even locally, were made over relatively slow serial lines. With Ethernet, a node could send data at a burst rate of 10 Mbps and a sustained data rate in the megabit range. But this could only be sustained within users of the LAN itself.

Once the LAN meets the wide area network (e.g., via a modem-equipped circuit), a severe mismatch between speeds occurs. How does a device equipped with a 10 Mbps interface know how to not overload a 9600 bps modem link? In Fig. 1, Host A can send to Host B at LAN speeds, but can only send to host C at slow speeds.

![Fig. 1. Funneling from a LAN to a WAN.](image)

One very crude technique is to rely on a simple acknowledgment mechanism to ensure that one packet is received before the next is sent. The device that connects the LAN to the WAN (e.g., a router) will necessarily have a queue for access to the WAN, which will induce delay and thus slow down the traffic to a manageable level, as senders...
wait for their packets to be acknowledged. But this is overkill: It does nothing to allow bandwidth to be efficiently utilized when there is substantial delay between the time a packet is sent and the time it is acknowledged. And in a multi-hop network, delays can be quite substantial compared to the time it takes to send a single packet. An analogy can be drawn to the old bisync protocol, which waited for each frame to be acknowledged, and did not perform satisfactorily over satellite circuits.

2.1. Sliding window protocols

To facilitate higher speeds, many protocols make use of a sliding window mechanism. This allows several unacknowledged packets to be outstanding at any time. Absent congestion, throughput increases along with window size until the window becomes so large that acknowledgments are received before a full window is sent. (This condition is sometimes called "filling the pipe"). Beyond that level, there is no point in using a larger window size. Most connection-oriented network layer and data link layer protocols use windows, but not connectionless protocols nor, of course, the Core Aspects protocol used by Frame Relay. In these windowless networks, the Transport layer, operating between end systems, typically maintains the sliding window.

When LANs were introduced, the use of then-extant sliding window transport protocols, such as early versions of TCP and DECnet's NSP, resulted in occasionally serious congestion problems. The scenario was simple. A given router had a finite amount of buffer space leading to its outgoing serial line. A few users would each send a window's worth of packets down the Ethernet. A buffer would fill, and some packets would be discarded. The receiver would request retransmission beginning from the missing packet, and the sender would oblige and send another full window beginning with the first missing packet. These retransmissions of course created additional load upon the network, leading to more dropped packets. This positive feedback loop could lead to what came to be known as congestion collapse, in which very little actual data got through.

Even non-LAN interconnections could lead to congestion collapse. We observed the same effect on a production network when a serial line's rate was raised from 56 kbps to 128 kbps. The higher speed line overloaded the 56 kbps circuit onto which most of its traffic was eventually routed, creating loss and a cycle of retransmissions.

Fortunately, the solution to congestion collapse turns out to be a simple one, called the dynamic window. This was invented for DECnet by Raj Jain and K.K. Ramakrishnan [TR72], and was later extended to TCP by Van Jacobson at Berkeley [JAC88]. (IBM independently invented a similar scheme.) In DECnet, the algorithm is referred to as Congestion control Using Timeouts at the End to end (CUTE). Jacobson's version, the "slow-start" algorithm, is now a mandatory part of TCP. Note that because DECnet and IP do not guarantee the sequential delivery of packets, receipt of an out-of-order packet does not imply that one has been lost, so a timeout is required.

The dynamic window is fundamentally simple. Whenever a packet is dropped (as noted by a timeout or, in the case of LAPD+ over Frame Relay, a gap in the received sequence) the window is slammed shut, to the size of one packet. This seems severe at first. But it makes sense because at the time the packet is dropped, the router that dropped it must have had exactly zero buffers left, or it wouldn't have been dropped. So even a single packet will only cross the network after a buffer has opened up.

Once a single packet has been successfully acknowledged, the window size is incremented. Two packets may now be sent before waiting for acknowledgment. One round-trip (window turn) later, if these are both successfully acknowledged, then the user may again increment the window size. This continues until another packet is dropped, or until the pipe is full or the computer is not capable of sending any faster.

The behavior of a connectionless network in which each and every user obeys the dynamic window discipline is quite predictable. The effective throughput for each user tends to follow a sawtooth pattern, rising slowly, and falling rapidly once congestion is inferred. (See Fig. 2.) In TCP, the rate of window size increase is slowed down at half the level at which the last dropped packet is noted, which tends to produce a "chipped sawtooth" pattern with somewhat longer periodicity.
3. Congestion avoidance

While it is stable over time and on average provides high throughput, the dynamic window alone does not provide the best possible results. In particular, it suffers from longer than ideal transmission delays, as the buffers in each interworking unit tend to oscillate between nearly full and empty. When the buffers are nearing full, queuing delays are long. Throughput per se does not suffer, but delay-sensitive applications become perceptibly slower. If delay could be reduced and throughput maintained, then then network could be optimized for power, which is computed by $P = T^\alpha / D$, where $T$ is throughput, $D$ is delay, and $\alpha$ is a weighting factor (usually 1) to allow power to be optimized for either higher throughput or lower delay, as the application requires.

This leads to the need for a protocol for explicit congestion avoidance. The goal of such a protocol is twofold. One is to reduce the frequency with which packets are actually lost, so that the implicit dynamic window congestion recovery mechanism becomes a backup rather than a first line of defense. The second is to reduce the average queuing delay throughout the network, without reducing average throughput. These must be accomplished in an environment in which the boundary nodes at the edge of the Frame Relay network are unaware of the actual end-user demands upon them, since there may be any number of Transport connections multiplexed across each physical link, and any number of hops at either side of the boundary nodes. And it must be accomplished in a manner which recognizes the hugely dynamic and variable environment of the customer's own network, which may include circuits ranging from 2400 bps dial-up modems up to and including FDDI LANs running at 100 Mbps — a dynamic range of about 40,000 to 1!

3.1. Potential methods

Several methods were explored by Jain, Ramakrishnan and Dah-Ming Chiu [JAIN88][RAMA90] before the explicit binary feedback technique was settled upon. One fairly obvious approach was to have the network send source quench messages to the senders of packets that encounter congested resources. Source quench packets were even defined within the Internet Control Message Protocol, without clear definition of what to do when received. (Their use is no longer recommended.) The problem with source quench is also fairly obvious: They add additional traffic to the network at a time when it is already congested.

In the experimental days of the ARPAnet, congestion status was sent between nodes as part of frequent routing exchange messages. This allowed routing to be recalculated to adjust to load. While this approach sounds intuitively useful, in practice it led to instability as traffic oscillated between alternate paths. In addition, frequent routing exchanges produced far too much overhead traffic. Frame Relay, of course, does not depend upon dynamic routing exchanges, so this approach would not have been applicable anyway.

Another approach is to send probe packets into the network to determine its current state. This too adds traffic; the more often probes are sent, the more traffic is added. But less frequent probes are less accurate. What if, instead, probe packets could be piggybacked onto other packets? This leads directly to the approach that was settled upon.

3.2. Binary feedback principles

In explicit binary feedback, every packet is a probe packet. As each packet traverses each node, it probes the instant congestion status of that node. Only two congestion states are noted, above ideal load (congested) and not above ideal load (not congested).

Because there are only two states, only one bit is needed for this probe function (hence the "binary"). Information theory dictates that the maximum useful information (entropy) is received from any bit when it is set half the time. Thus the maximum useful information about network condition is obtained when, during periods of load, the probe bit is received set half the time and clear half the time.

This leads to a definition of ideal load that is generally quite conservative. A node need not be in imminent danger of losing packets to be "congested"; it need merely have more load than that which provides the desired balance between short delay time (small buffer utilization) and high throughput (larger buffer utilization). When the node is above the optimal point, it sets the bit; when it is below or at that point, it does not set the bit. No node may clear the bit; if any node along the path sets it, then it must arrive set.

Because this is a feedback scheme, certain rules apply. Feedback, to have maximum impact, must be neither too fast nor too slow. If it is sent and used too quickly, then oscillation may result. (That can be simulated by holding a public address system's microphone too close to the speaker.) If it is too slow, then it will not be effective. At the speeds at which most Frame Relay networks operate, or for that matter any wide area network using today's technology and network speeds up to around 2 Mbps, timeliness is not a problem. Instead, the feedback is filtered to add a degree of hysteresis and prevent oscillation.

Once a packet arrives at its destination (a transport layer entity), the congestion bit is fed into a filtering algorithm to create a time-averaged congestion metric. The basic filtering time period is the round-trip delay across the network. (In practice, this can be approximated and imple-
mented by using a window turn; a packet's acknowledg-
ment comes at least one round trip later.) During this in-
terval, the number of packets in which the congestion bit
arrived set is compared with the number in which it ar-
rived clear.

If at least half of these bits are set, then the path is con-
sidered congested; if fewer, then the path is considered
uncongested. This 50% threshold is not arbitrary: Infor-
mation theory dictates that the maximum information
content of a bit (entropy) is achieved when that bit is set
half the time. This is semantically quite different from,
say, a source quench packet, which would likely be sent
only during periods of severe congestion.

\[ \text{Knee} \quad \text{Cliff} \]

Throughput

Load

Response time

Load

Power

Fig. 3. Network performance as a function of
load. Broken lines indicate performance with
deterministic service and interarrival times.

[JAIN88]

Having determined the (binary) congestion state of the
path across the network, the user then adjusts the offered
load. (In practice again, this often translates into a win-
dow size adjustment.) If the network is congested, then
the user should multiplicatively reduce the offered load
by a small margin; i.e., by reducing the offered load to
7/8 of its previous value. If the network is not congested,
and the user's window size has not "filled the pipe", then
the user may safely increase the offered load in an addi-
tive (not multiplicative!) fashion; i.e., by increasing the
window size by 1 packet, or the offered load by a con-
stant. In the case of rate-based frame relay, that constant
may be a fixed fraction (i.e., 1/16) of the negotiated rate.

Note the use of multiplicative vs. additive adjustment. If
the user were to increase its rate multiplicatively, then
users with higher rates would get larger increases than users
at lower rates. A similar problem would occur if the de-
crease were additive: Small users would drop to too low a
value, too quickly. Additive increases and multiplicative
decreases, on the other hand, will quickly converge and
provide the most "fairness".

3.3. Network behavior

How is the network to determine its own level of conges-
tion? In this too, some degree of hysteresis is required.
But the network switch can't easily apply the concept of
round-trip delay or window size, as it serves many users
whose round trip delay varies widely. We found that the
best results occur when the network node determines its
congestion state by finding the average fill level of its
output buffer over a period defined by regeneration cy-
cles.

A regeneration cycle begins when a queue drops in size
to 0. In any packet-switched network, circuits will have
some idle time, when the buffer is empty. When a packet
arrives, the queue size is 1; a cycle continues until it
drops back to 0. If a packet is being sent, then by defini-
tion it is in the buffer, so the queue size cannot be 0. In
setting the congestion avoidance bit, the average queue
size is computed going back in time until the beginning
of the regeneration cycle preceding the current one. This
means that there will, by definition, be some idle time
(queue=0) within this period. If this average, at the time
the packet is sent (reaches the head of the queue), exceeds
a stated threshold, then the packet is marked "congestion
encountered".

We have found that an average queue size of 1 produces a
good balance between interactive performance (short
queues) and high throughput (longer queues) when ap-
plied to typical DECnet traffic. In the case of Frame Re-
lay networks, larger queue sizes may be more appropri-
ate, especially for high-speed backbone facilities. But this
can be an adjustable parameter, with which the network
may influence the load placed upon it by users.

When a packet traverses such a congested node, the ex-

plicit congestion notification bit is set. No subsequent
node may clear it. The received value of this bit is the
logical "OR" of the state of all nodes that relayed the
packet. This final value is fed into the end system's deci-
sion process.
4. Applicability to Frame Relay

This binary feedback scheme was originally developed for connectionless networks, in which the congestion avoidance bit exists within the Network layer header and flow control exists within the Transport layer. But this assignment of functions to layers is not rigid; there is no reason that it cannot operate within the Frame Relay context.

To be sure, there are two rather different approaches by which users may cooperatively perform flow control with Frame Relay. One is to directly run an edge-to-edge protocol that deals only within the bounds of Frame Relay. For example, the LAPD+ protocol provides the Data Link layer service across Frame Relay, and incorporates the Dynamic Window. The other approach is to send unsequenced frames across the Frame Relay, treating it essentially like a connectionless LAN, and allow higher layer protocols (i.e., Transport) to handle the flow control on an end-to-end basis. In this latter case, flow is managed by the actual end systems, which may be some hops away from the Frame Relay network and blissfully unaware of its existence!

The edge protocol approach is probably best suited to simple TEs, like ISDN telephones and terminal adapters that attach a single terminal (or personal computer emulating one) to an ISDN. These users do not generally require a full "seven layer" stack, so a simple solution is preferred. Another possible advantage to an edge protocol is to maintain good response time: If the lostiest link in a multi-hop network is a frame relay, then recovering from losses at the frame relay level will reduce the frequency with which more time-consuming end-to-end recovery will be needed.

But it is not possible to use the Core Aspects protocol's Forward Explicit Congestion Notification (FECN) bit, which provides the explicit feedback as described above, directly with LAPD+, because LAPD+ keeps its only window in the transmitter, and the FECN bits are counted by the receiver. The Backward Explicit Congestion Notification (BECN) bits, set on packets being returned to the sender on congested virtual circuits, may be usable with the LAPD+ window, but this approach was not selected for DECnet. Instead, we chose to use end-to-end controls in conjunction with the Transport layer. This is considerably simpler and takes full advantage of the capabilities of the Network and Transport layers.

This approach is also more accurate than using backward feedback, because the effect of the FECN bit is always returned to the sender (by Transport) one round-trip after the congested packet is sent. A BECN bit, on the other hand, will be returned to the sender at a different interval after the packet is sent, depending upon where the congestion is located. If the congestion is near the sender, then the feedback is likely to be returned before the packet is even acknowledged; if it is near the receiver, it will take considerably longer. The consistent delay in FECN allows us to discard the bits in the first window received after an adjustment and count the bits only in the second window turn (allowing some time for the network to react to the adjusted load). This would not be possible with the variability in backwards feedback.

4.1. Signaling the OSI Transport layer

The OSI Connectionless Network layer protocol is encapsulated within HDLC Unnumbered Information (UI) frames. If the received UI frame has the Core Aspects FECN set, then the Congestion Encountered bit in the Network layer header is set. For each virtual circuit, the Transport layer then compares the number of packets received with this set with the number received with it clear, to arrive at a decision as to whether the transport layer window should be made larger or smaller.

This is effective because the Transport protocol operates under receiver control. The transmitter may only send the number of packets for which credit has been granted. Typically this is done to reflect the amount of buffer space available in the receiver, to ensure that its own memory does not overflow. But if the receiver detects that the network is congested, it can use the same mechanism to reduce the rate at which it grants credits, and the transmitter will slow down just the same.

In the event that a packet is lost, the ISO Transport Protocol Class 4 allows the receiver to selectively request that it be retransmitted. (This is more efficient than the Go-Back-N recovery technique used by LAPD+.) Such an event also implies that congestion has reached the stage in which some network resource, whether in the Frame Relay or in some other subnetwork between source and destination, has exhausted its buffering capability. In this case, a separate window in the transmitter immediately shrinks to the size of one packet.

Three separate dynamic limits are now being placed on the rate of transmission. One is receiver buffer memory, and the second is the receiver congestion avoidance window. The credit sent from the receiver is the lower of these two values. The third limit is the transmitter window, which only shrinks as a result of packet loss, and thus provides recovery from serious congestion events.

The congestion recovery window in the transmitter grows twice as quickly as the congestion avoidance window in the receiver. The former grows by one packet for every lossless window turn. The receiver window rises by one packet no more often than once every two window turns: The percentage of congested FECN bits is measured during alternate window turns, changed, and its effect is noted (with no change to the window size) on successive window turns. Following a congestion event that results
in packet loss, the actual transmission rate rises relatively quickly so long as the transmitter window is the smaller one; it then rises more slowly once the receiver window becomes the gating factor.

The Transport layer protocol, of course, is unaware of the presence of Frame Relay. In a connectionless network, occasional packet loss is the norm; congestion avoidance and recovery are a way of life. The congestion control mechanisms are, in effect, a servo mechanism that dynamically regulates the flow of traffic across an arbitrarily wide range of speeds and arbitrarily complex network topologies. Frame Relay is a relatively straightforward option, fitting into the middle range of bandwidth. It is typically faster than the modems used across analog circuits still prevalent in today’s networks, but slower than Local Area Networks. While it induces delay, it adds less than X.25, which is already widely used as a subnet.

4.2. Enforcing uniformity of behavior

A Frame Relay network, like a connectionless network or any other network that does not use positive flow control, depends upon the cooperation of its users. If some users reduce their offered data rate in the face of congestion while other users do not, the former users will be altruistically giving up their share of the network to the latter. Since altruism is rarely rewarded, and certainly not here, the temptation is for users to be on the receiving, rather than the giving, end of such largess.

If a network is entirely the property of one customer, then that customer has only itself to blame if some of its computers crowd out others. Private Frame Relay networks fall into this category. In the case of public networks, such a lack of fairness can be more problematic. A customer whose computer follows a more aggressive policy, for example one which shrinks its window more slowly or grows it more quickly, could in effect be denying service to others.

Frame Relay networks are not capable of directly enforcing any sort of window-based rate pacing algorithm, as the window is "hidden" somewhere within the upper layer payload. But networks are entitled to look at the FECN and BECN bits, may monitor data rates on its virtual circuits, and can discard frames as they see fit. This permits networks to both police (occasionally seek out violations) and enforce (routinely prevent violations) compliance with congestion guidelines.

One option to ensure fairness is for the network to continually monitor the data rate, averaged over time, for each virtual circuit, and to discard packets that exceed a pre-negotiated limit. This is an optional network procedure facilitated by the Discard Eligible (DE) flag in the Core Aspects header. If the user exceeds its negotiated rate, then the DE flag may be set by the access node; suggested network nodes may then discard frames whose DE flag is set in order to make room for frames whose DE flag is clear. This is transparent to the user. The access node may also discard frames from users who flagrantly violate the negotiated rate.

A cooperating user will, of course, severely reduce its data rate upon detection of frame discard. That will reduce the probability of further frame discard. A simple rate-based source enforcer does not respond to the dynamic state of the network. But the FECN and BECN bits can be used by the enforcer to change both the DE threshold and the access node discard threshold. If most of the frames on a virtual circuit indicate congestion, and the user is supposed to slow down, then the source enforcer may lower its thresholds accordingly. These may then rise back to negotiated levels at a rate that approximates the behavior of the dynamic window. Since a window turn corresponds roughly to one round-trip across the network, the network can use the end-to-end delay parameter to set the interval at which it adjusts its thresholds.

Enforcement and to a lesser extent policing add complexity to network switching elements. Thus their use may be tempered by the experience of the network operator. Frame Relay is designed to be fast and cheap, and it does this by omitting most of the controls found in X.25. With the cooperation of users, and the widespread acceptance of explicit binary feedback, Frame Relay can be a successful and cost-effective alternative to traditional data switching techniques.

References


