EUROCONTROL

ATNP/WG2/
WP/260
15 April 1999

AERONAUTICAL TELECOMMUNICATIONS NETWORK PANEL

WORKING GROUP TWO

Brussels, 22.4.96-26.4.96

Proposed Guidance Material for Congestion Avoidance in the ATN Internet

Presented By Henk Hof

Prepared by Tony Whyman

SUMMARY

The draft ATN Internet SARPs now includes a specification for the implementation of a receiver based Congestion Avoidance algorithm. This working paper provides draft Guidance Material in support of this specification.

DOCUMENT CONTROL LOG

SECTION	DATE	REV. NO.	REASON FOR CHANGE OR REFERENCE TO CHANGE
	15-apr	Issue 1.0	

TABLE OF CONTENTS

1. Introduction

1.1 Scope

This paper provides draft Guidance Material for Receiver based Congestion Avoidance in the ATN Internet. Material from **[1]** has been used in the preparation of this text.

1.2 References

1. WG2/WP254 Comments on the Congestion Avoidance Algorithm proposed for the ATN

2. Congestion Avoidance in the ATN Internetwork

2.1 Network Congestion

Congestion is a phenomenon experienced by a Router in an Internetwork when the queuing delays through that Router exceed the maximum acceptable limit. In such a situation, the end-to-end transit delay is likely to exceed the maximum acceptable for the internetwork's users. In the extreme case, a congested router, due to lack of buffer space, may not be able to accept incoming NPDUs at the rate that an adjacent router is trying to send them, and is hence forced to discard lower priority NPDUs, or those near the expiry of their lifetime, in order to make way for higher priority NPDUs.

Congestion is not a problem for an internetwork. Congested routers can simply discard NPDUs when they start running out of buffers. However, it is a serious problem for the users of the internetwork. Congestion first results in an acceptably long transit delay. However, if network users assume that the lack of arrival of an end-to-end acknowledgement is due to packet loss, rather than simply an unexpectedly long delay in the network, then they can retransmit such unacknowledged packets, thus adding to the load on the network.

In fact, a catastrophic degradation in transit delay and throughput can be observed in a congested network. First the network becomes congested, then users start retransmitting, making the network even more congested, resulting in more retransmissions, and so on, until the point is reached where only insignificant amounts of data can be transferred. It is therefore vital that Congestion Avoidance mechanisms are put in place in any internetwork, if it is not to be perceived as unstable and unreliable.

2.2 Possible Techniques

In a connectionless internetwork, Congestion Avoidance has to be a co-operative activity in which a major part is played by the users of the network. Successful operation of the network depends on its users being "good citizens" and reducing the load placed upon the network once the onset of congestion has been determined.

In general, any suitable Congestion Avoidance technique must be able to control overload situations in the underlying network in such a way that data transfer is performed as efficiently as possible. To be acceptable, the adopted Congestion Avoidance technique must satisfy the following goals:

1. High throughput (in bit/s), together with a small end-to-end transit delay, should be experienced by network users.

2. A small buffer load within the traversed routers should be achieved.

3. The probability of packet loss should be minimal.

In pursuit of these goals, two candidate algorithms were initially investigated during the development of the ATN Internet SARPs. These were a sending transport entity back-off algorithm, similar to the Van Jacobsen Slow-Start algorithm that is widely used in the TCP/IP Internet, and a Receiving Transport Entity Congestion Avoidance algorithm.

Although widely used, the former was rejected. The Slow-Start algorithm probes the network until congestion occurs, when the transport entity backs off and then proceeds to probe again. It is effective when congestion is a rare event, and avoids catastrophic congestion occurring, but is inefficient on a heavily loaded network, as that network is regularly forced into a congested state during the regular "probes". In a mobile network, such as the ATN, there is also considerable scope for the Slow-Start algorithm to be confused by a mobile system changing its point of attachment. The resulting packet losses will be interpreted by the sending transport entity as an indication of a congested network, forcing a back-off state and hence a resulting in a lowering of throughput.

On the other hand, the chosen algorithm relies upon indications received from the network layer (i.e. the CE-bit in an NPDU Header) in order to determine when the network is approaching a congested state, and adjusts the advertised credit window in response. This has the advantages of avoiding the continued probing that is characteristic of the Slow-Start algorithm, and of remaining unaffected by a mobile system changing its point of attachment. It therefore appears to give a significantly better throughput in the aeronautical environment.

2.3 Receiving Transport Layer Congestion Avoidance

2.3.1 Overview

The Receiving Transport Layer Congestion Avoidance algorithm depends on the "Congestion Experienced" (CE) bit that may be included in an NPDU Header. This bit is set initially to zero by the End System that creates the NPDU. Should the NPDU pass through a Router, on its journey through the internetwork, that is either congested, or is nearing the point of congestion, then the CE-bit is set to one by that Router.

When an NPDU is received by the destination End System, it can therefore readily inspect the CE-bit and determine if the NPDU experienced congestion anywhere on its route.

This is a simple mechanism for determining the congested state of an internetwork, and does so without generating additional network traffic. This is important, as a network reaching the point of congestion should not suffer additional traffic, just because it is congested!

When the receiving transport entity gets an NPDU with a CE-bit set to one, it is not required to take immediate action - indeed, it should not do so, as such an isolated event may well be transitory. However, if enough NPDUs are received with a CE-bit set to one, during a suitable sampling period, then it must take action to tell the sending transport entity to slow down and reduce the load it is putting on the network. This is because when NPDUs start to be regularly received with the CE-bit set, then it is indicative of a network that if not already congested, is starting to become so.

The way the receiving transport entity tells the sending transport entity to slow down, is to reduce the advertised credit window. Whenever a received TPDU is acknowledged, an AK TPDU is sent back to the sender that includes the sequence number of the most recently received TPDU and gives permission for the sender to send another *n* TPDUs. Normally, the objective is to acknowledge received DT TPDUs in a timely manner to ensure that the sender never gets into a situation whereby it no longer has permission to send any more DT TPDUs (i.e. that it runs out of credit). The sender is then able to transmit data as fast as it can.

Normally, *n* is set large enough for this to be the case. However, when congestion is detected, if the receiving transport entity sets *n* to a smaller value, the sender will start to occasionally run out of credit, there will be times when it cannot send any DT TPDUs, and hence the load on the network is reduced. Thus the sending transport entity can be readily told to slow down simply be reducing the value of *n*.

Later on, if a smaller proportion of packets are received with the CE-bit set to one, then *n* can be safely increased again, until congestion is once again determined. On a congested network, this algorithm results in a small oscillation around the ideal data transfer rate, while never pushing the network into a congested state. A high throughput with the minimum of transit delay is thereby achieved, without forcing the routers to discard packets as part of a "probing" process. Our goals for a Congestion Avoidance Algorithm are thereby met.

2.3.2 Determining the Onset of Congestion

In line with the definition given earlier, a router can be considered congested when the queuing delays imposed by a transit through it exceed a certain threshold. A useful metric for congestion can therefore be gained from a simple inspection of the length of the outgoing queue, when a forwarded packet is queued for transfer to another system. If the queue exceeds a certain length then the CE-bit should be set to indicate that the queuing delay is excessive i.e. congestion has been experienced. However, what is an appropriate queue length (i.e. the threshold) to determine when the CE-bit is to be set?

When specifying a queue threshold, it is necessary to take into account what it is intended to do with this signal. The final goal is to achieve a data transfer service with fairly good user visible performance (i.e. low end-to-end delay, high throughput), without producing too high a buffer load (so the global network is operating stable). If the buffer load found in any output queue is very large, it runs the risk of packet loss, which will trigger packet retransmissions. These in turn will increase the end-to-end delay of the data transmission (since packet loss first has to be detected and recovered, before normal data transmission can continue) and thus will also reduce the throughput visible to the user. Finally, packet losses put an additional burden on the network, since the lost packet will already have (uselessly) traversed part of the network, before it gets lost.

Since high throughput and low end-to-end delay are competing goals, L. Kleinrock proposed in his standard work on queuing systems to optimise the "Power" of a connection, which he defined as

$$\text{Power} := \frac{\text{Throughput}}{\text{Delay}}$$

This measure has served well since its introduction, and is widely used within network optimisation. By adapting that goal to the problem considered here, we have to derive a threshold value for the output queue load such that the Power of the system is maximised.

To derive an appropriate queue threshold value, we consider an output queue together with its outgoing link as a M/M/1 queuing system (exponentially distributed inter-arrival times, exponentially distributed service times; see also Figure 2-1).

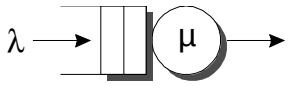

Figure 2-1 A queuing system

Packets arrive at a server with arrival rate λ, where they eventually get queued if the server is currently busy. Packets are fetched from the queue by the server, which forwards packets at a rate of μ. The system is in a stable state only if packets do not arrive faster than the server can forward them, i.e. if and only if $\lambda < \mu$.

Such a queuing system is referred to as an M/M/1 system, and for such an M/M/1 system, the average time a packet spends in the system is given by

$$E[T] = \frac{1}{\mu - \lambda} \quad (1)$$

The throughput of an M/M/1 system is equal to λ, if the system is operating in a stable state (i.e. one can never receive a higher throughput than the server forwarding rate μ, but the server is also not able to forward packets faster than they arrive).

From the above, the Power of a M/M/1 system thus can be derived to be

$$\text{Power:} = \frac{\text{Throughput}}{\text{Delay}} = \frac{\lambda}{1/(\mu - \lambda)} = \lambda \cdot (\mu - \lambda) \quad (2)$$

This measure is maximised if the following condition holds:

$$\lambda \equiv \frac{\mu}{2} \quad (3)$$

The average number of customers found in a M/M/1 system is given by

$$E[N] = \frac{\lambda}{\mu - \lambda} \quad (4)$$

which, for λ as given in (3) to optimise the power, finally evaluates to a value of 1 packet. If the output queue threshold is thus set to 1 packet, every system will try to operate at a point of maximum power, i.e. offering a high throughput to the user, while also making sure that the end-to-end delay (e.g. for short messages exchanged between communicating entities) is kept reasonably small.

Although it has been suggested that a number greater than one is appropriate low bandwidth data links, consideration of the above shows that there is no justification for this. A value larger than one simply implies that longer queuing delays are tolerated with clear downside implications for throughput (e.g. by requiring a longer retransmission timer, reducing the rate at which AK TPDUs can be sent, etc.).

However, this is not to say that special considerations do not apply to air/ground data links. The queuing model presented in Figure 2-1, and the associated argument assumes that all outgoing queues from a given router are independent. This is not true for a network such as AMSS, where a single transponder is shared for communication with all aircraft. Although the Air/Ground Router servicing an AMSS data link will see a separate outgoing queue for each aircraft, the reality is that they are all constrained by a common uplink queue. In such cases, the number of packets on the outgoing queues should be summed up and the CE-bit set when the total packets queued for uplink over the same transponder is greater than one.

2.3.3 Reporting Congestion Experienced to the NS User

Congestion is experienced by an NPDU, while it is an NSDU that is passed to the NS-User as part of an N-UNITDATA.indication. In many, if not most cases, there will be a one to one relationship between NPDUs and NSDUs. In such a case, there is little problem in reporting Congestion Experienced, and, as additional information to the N-UNITDATA.indication, the Network Layer can pass an indication that the NSDU reported congestion experienced on its route from the sender.

However, this leaves open as to what happens when an NSDU is segmented into two or more NPDUs, some of which may experience congestion, while others do not. Possible strategies for the network layer are to:

1. to indicate to the NS-User both the total number of NPDUs received for a single NSDU, and the number of NPDUs received having the CE flag set to the transport layer;

2. to merge the CE flags received by bitwise ORing all values. Thus, if a single NPDU had the CE flag set, congestion will be indicated to the NS-User.

3. to merge the CE flags received by bitwise ANDing all values. Thus, only if all NPDUs had the CE flag set, congestion will be indicated to the NS-User.

4. to only forward the CE flag setting of the last NPDU received during reassembly of an NSDU to the local transport layer.

Strategy (1) is the preferred strategy. This is because it gives the NS-User the maximum amount of information on which to base a decision. All the alternatives hide information from the NS-User, and there is little value in doing so.

2.3.4 Credit Window Management by the Receiving Transport Entity

The receiving transport entity monitors incoming TPDUs and determines whether or not congestion was experienced by the TPDU during its transit through the internetwork. If, during some sampling period, congestion was experienced by enough TPDUs, then the effective credit window is reduced by multiplying it by a reduction factor β. Otherwise, if the credit window is currently less than the a value which will permit maximum throughput, then it may be increased by adding an integral value δ. Initially, the credit window is set to a low value (e.g. two). The algorithm then ensures that it increases until either maximum throughput is achieved or, congestion starts to be experienced, when the credit window oscillates about the optimal value. Note that starting from a lower value (i.e. one) has a downside in that a credit of one demands two AK TPDUs for each DT TPDU transferred.

Only DT TPDUs are monitored during a sampling period. This is because only DT TPDUs are subject to credit management. Other TPDUs types, such as expedited data or acknowledgements are not subject to credit management, and therefore no feedback can gained by monitoring them to see if any restrictions on the credit window are working in respect of reducing network congestion.

Furthermore, once a sampling period has been completed, and a new credit window determined, no more sampling should be undertaken for a period equal to the estimated Round Trip Time (RTT). This is because any DT TPDUs received during this period will have been subject to the previous credit window. Only once the RTT has elapsed, can it be assumed that the received DT TPDUs are subject to the new credit regime and hence its effect on the network state can be reasonably determined.

The reason why a “freeze period” is necessary can be readily seen from the following example.

Figure 2-2 depicts a sender transmitting data towards a receiver. Each packet is indicated by a line going from the sender to the receiver. Transmission of a packet through the network takes a certain amount of time, represented by the slope of the line (time proceeds from top to bottom).

Initially, the transmission is performed in this example with a window size of 8. It is also assumed that the network is currently overloaded, so the receiver will see CE flags being set, and reported to the Transport Entity by the Network Layer.

At time t_1, the receiver decides to ask the sender to reduce its load, in order to remove the overload found in the network. The sender is informed about this decision using an AK TPDU, transmitted from the receiver to the sender (indicated by the dashed line in Figure 2-2).

Once this indication is received by the sender, the sender reduces its window to the value advertised by the receiver. For the scenario considered here, it is assumed that $\beta = ½$, to better visualise the

operation. Thus, the window W is reduced to W_1= 8 x ½ = 4. Afterwards, the sender is transmitting with a smaller load, indicated by a greater spacing of the packets.

As can be seen from the figure, immediately after the decision to reduce the advertised window, the receiver will continue to get packets still transmitted with the old window. These may also have their CE flag set, since the sender is not yet aware of the decision to reduce the window, and still is transmitting with the large window. It takes approximately one round trip time, until the first DT TPDU transmitted at the lower load (i.e. with the reduced window size) arrives at the receiver. Note that within this time interval, W_0 packets will be received that still have been transmitted using the old window size.

During the next round trip time (starting at time t_2), DT TPDUs being sent using the reduced window size, arrive at the receiver. Assuming that the load is now small enough, there will be no more congestion within the network. In consequence, these packets will not have their CE-flag being set. The receiver will thus see another 4 packets without the CE flag set. After the second RTT (i.e. at time t_3), the receiver will make a new decision how to modify the advertised window size.

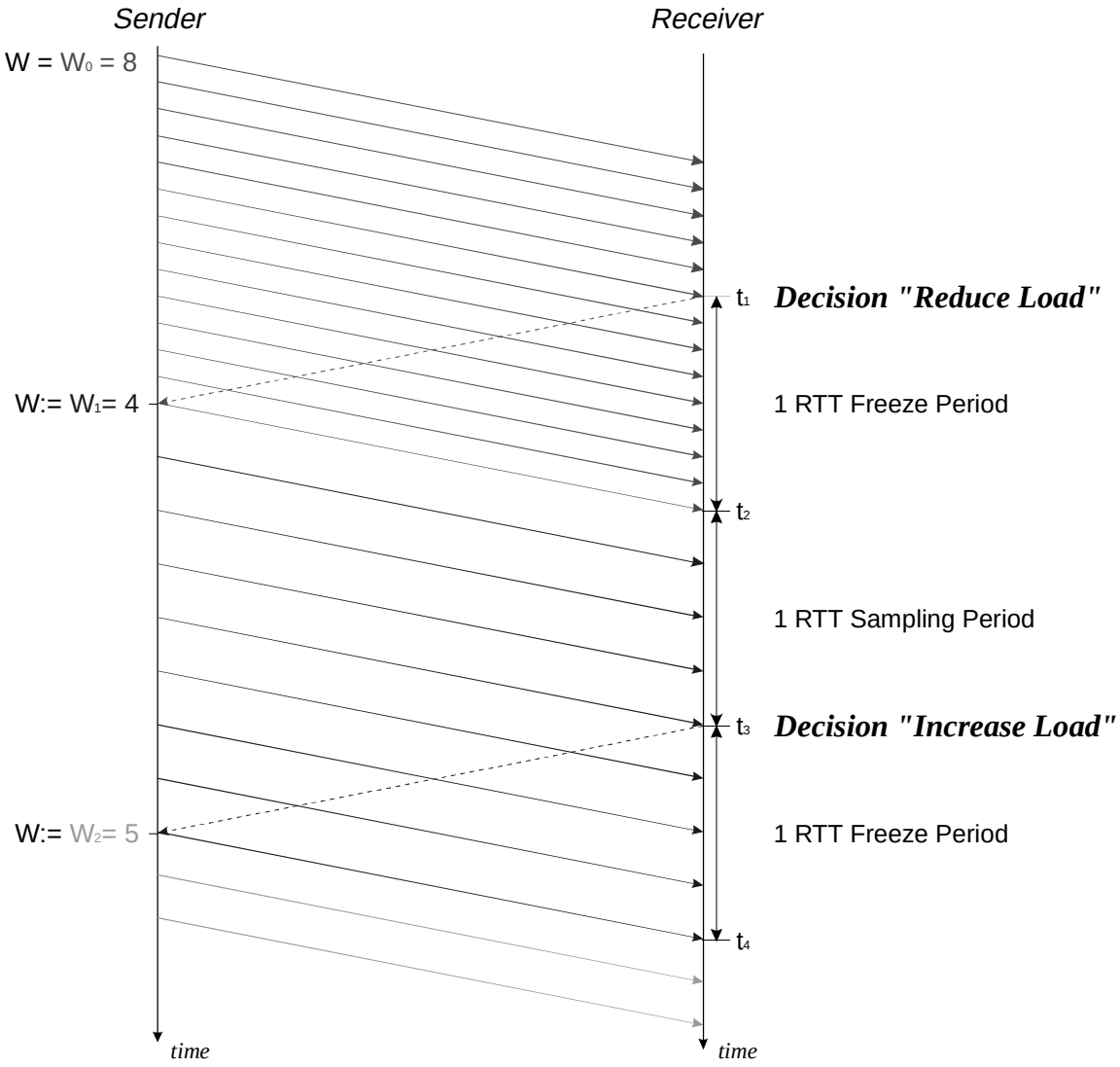

Figure 2-2 Window Adaptation over Time

2.3.5 The Congestion Avoidance Algorithm

In the ATN Internet SARPs, the Congestion Avoidance algorithm is presented as a set of requirements, following the normal style for SARPs. It is represented here in a 'C' code format, in order to make the algorithm more readily understandable to implementors.

Firstly, to support the Congestion Avoidance algorithm, each connection keeps a number of state variables, defined and initialised as follows:

```
int  n_DT    = 0;          // number of DT-TPDUs received
int  n_total = 0;          // total number of CE signals received
int  n_CE    = 0;          // number of active CE signals received
int  W_old   = 0;          // previously advertised window size
int  W_new   = W0;     // newly advertised window size
bool sampling= TRUE;       // are we currently sampling CE-flags?
```

Note that a new connection starts advertising an initial window size W_0 (as defined in SARPs text 5.2.6.3) to its peer. This is reflected in the initialization of variable 'w_new'.

Whenever a TPDU is received from the network layer, the routine *CongestionAvoidance()* is called with the congestion information received from the network layer forwarded to it. This routine performs the congestion avoidance algorithm, and updates the state variables as follows:

```
CongestionAvoidance(bool dt_TPDU, int nTotal, int nCE)
{
  // dt_TPDU  - flag indicating whether a DT-TPDU had been received
  // nTotal   - total number of NPDUs forming that TPDU
  // nCE      - number of NPDUs forming that TPDU that had their CE flag set on
  //            reception

  if (dt_TPDU) n_DT++; // count total # DT-TPDUs received
  n_total    += nTotal;    // count total # signals received so far
  n_CE   += nCE;       // count # active signals received so far

  if (n_DT > W_old) {
         // received enough DT-TPDUs, phase is completed
     if (sampling) {
            // was in sampling phase; compute new window and advertise
         if (n_CE > lambda * n_total) {
            W_new *= beta;
         } else {
            W_new++;
         }
         AdvertiseWindow(W_new);
         sampling= FALSE;
     } else {
            // was not sampling; just switch to sampling phase
         W_old   = W_new;
         sampling= TRUE;          // now entering sample phase
         n_total = n_CE= n_DT= 0; // reset counts
     }
  }
}
```

Note.:'lambda', 'beta' and 'W0' are parameters defined in the SARPs text; see section 5.2.6.3: "Recommended algorithm values".

2.3.6 Sending Transport Entity Procedures

No specific features are required of the sending transport entity, in order to support this Congestion Management algorithm. It is only required to implement normal behaviour with respect to the handling of AK TPDUs and the utilisation of the received credit window.

However, implementors should note that commercial implementation of the transport protocol may often include "transport layer backoff" procedures similar to the van Jacobson Slow-Start algorithms. Implementors are strongly advised to remove such a feature from the implementation prior to it being deployed on the ATN. The backoff procedures is not required for congestion management and is likely to detect false indications of network congestion, when a mobile system moves its point of attachment. This will result in reduced throughput, and implementations that include the backoff procedure will be perceived as being slower and giving poorer performance than those that do not.

2.3.7 Known Limitations

2.3.7.1 Fairness

It is known from previous research in the area of Congestion Management algorithms, that the adaptation of a window (instead of the transmission rate) is likely to cause problems if competing users have different path lengths (i.e. round trip times). Such a situation is shown in Figure 2-3.

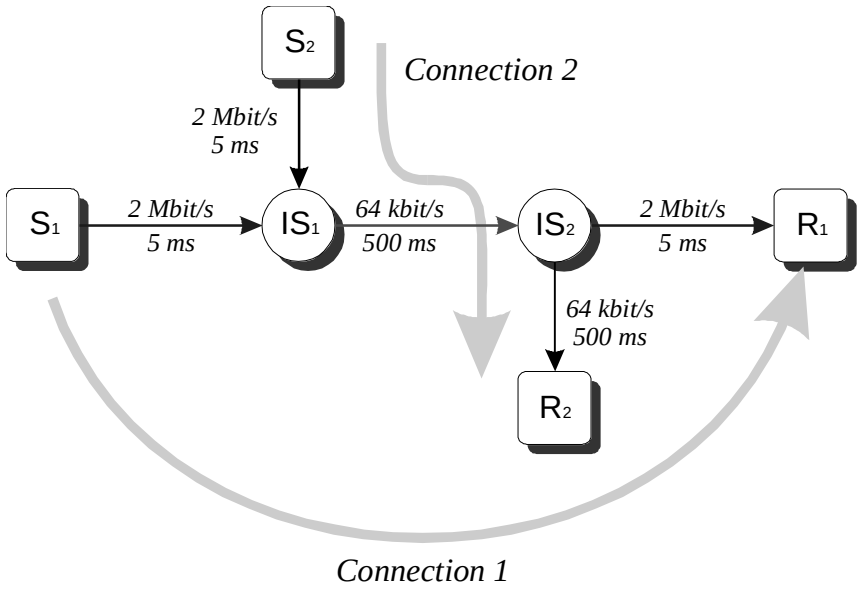

Figure 2-3 Fairness among competing Users

The problem is that the specified Congestion Avoidance will tend to result in approximately equal credit windows for all transport connections through the congested node. However, throughput depends not just on credit window, but also on the Round Trip Time. Once credit windows become restricted below the point at which greatest throughput is achieved, a transport connection will experienced a lower throughput than another with the same credit window and a shorter Round Trip Time.

It may be possible to balance throughput by varying the value of β taking into account the Round Trip Time. However, this requires network wide co-ordination to be effective and is only then useful with large window sizes. This limitation therefore appears to be a feature which has to be accepted.

2.3.7.2 Two-Way Traffic

Another well-known problem of many Congestion Control algorithms is caused by traffic along the reverse path. If data packets are transmitted along the reverse path, they will keep the intermediate system busy for some time. Acknowledgements arriving during that time will get queued, waiting for the IS to become available again. As soon as the system becomes free, these acknowledgements are transmitted back-to-back. This can have some adverse influence on the operation of the Congestion Control algorithm (e.g. leading to bursts of data packets emitted by one of the senders).

Figure 2-4 depicts this scenario.

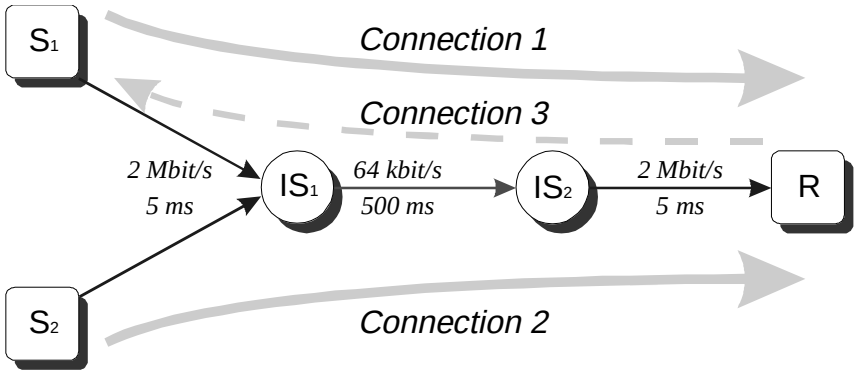

Figure 2-4 Two-way Traffic

2.3.7.3 A Credit Window of One is the Minimum

Like all Congestion Management algorithms, there is a point beyond which the algorithm cannot stop the network getting into a state of catastrophic congestion. In this case, this point is reached once all transport connections through a congested node have had their credit window reduced to one. After this point, the algorithm cannot reduce the load on the network any more and any increase in the load results in congestion, packet discards, re-transmissions and the network will become congested.

2.3.8 Conclusion

Congestion Avoidance is an essential feature for any internetwork. The specified algorithm appears to be the best for the ATN and achieves the best throughput while avoiding congesting the network as part of its own operation. There is, however, a limit to its effectiveness. This limit point is well beyond the point at which the algorithm starts to give useful benefits. However, it still underlines the important of good network design and capacity planning in respect of ensuring that network performance is maintained. A good Congestion Avoidance algorithm is an essential defence mechanism. However, it cannot give you network capacity that does not exist.