Continuous framing mechanism for congestion control in broadband networks

Jau-Hsiung Huang and Piau-Chuo Tsao

In this paper, a new congestion control mechanism, *continuous framing*, for broadband networks is proposed. This mechanism provides bounded end-to-end delay to all delay sensitive traffic, and guarantees loss-free transmission, if necessary. Compared with the stop-and-go discipline, it removes the extra waiting time introduced by node synchronization and supports time frames of any length. Based on this control mechanism, two enhanced mechanisms, *delay sending* and *virtual tag*, are presented to achieve intra- and inter-connection multiplexing gains such that the bandwidth reservation required for each connection can be reduced significantly. Implementation issues are also discussed in the paper.

Keywords: broadband networks, congestion control, framing

For computer networks, the problem of congestion control has recently been the subject of extensive research[1-6]. Traditionally, acknowledgement-based control is exercised, which makes use of the information fed by the downstream or destination node to regulate the input traffic. In broadband networks, the propagation delay, when measured in terms of the service time of a packet, is much longer than that in narrowband networks; consequently, the acknowledgement-based control mechanism is not suitable.

In addition, window-based flow control and the first-come-first-serve policy can neither satisfy some performance requirements, such as bounded delay and loss free transmission, nor provide firewalls among connections. Therefore, some rate-based congestion control mechanisms have been proposed. These control mechanisms include virtual clock[7], stop-and-go[8-11], hierarchical round-robin[12], fair queueing[13], delay-earliest-due-date[14] and jitter-earliest-due-date[15]. The first three of the above are of particular interest in this paper, and their pros and cons will be discussed below.

The main purpose of virtual clock is to maintain the statistical multiplexing flexibility of packet switching while ensuring each data flow its reserved average throughput rate. A flow that exceeds its reservation will get a worse service, since its virtual clock rate will run faster, which effectively builds firewalls between data flows. However, a bounded end-to-end delay is not supported without a peak-rate-based bandwidth reservation and a proper traffic policing policy.

The stop-and-go policy guarantees services per connection with a bounded end-to-end delay and a loss free transmission. A bandwidth reservation for each connection is made by the admission control based on its peak rate requirement. In this policy, it is desirable to incorporate multiple frame sizes according to different delay requirements. However, the frame size of the longer ones must be integer times of that of the smaller ones. Furthermore, the time framing on incoming links and that on outgoing links have to be synchronized at each node. To implement this synchronization, extra synchronization delay is introduced per node. Additionally, statistical multiplexing among connections is not provided.

Hierarchical round-robin has similar pros and cons to the stop-and-go policy. With careful comparisons, the hierarchical round-robin and stop-and-go policies are the same in many areas, such as service discipline, bandwidth allocation, buffer requirement and delay performance[5].

Based on the above discussion, we are inspired to design a novel congestion control mechanism which can achieve the advantages of the above policies without their disadvantages. The new congestion control mechanism, *continuous framing*, not only provides loss free transmission and bounded end-to-end delay, but also achieves a statistical multiplexing gain among connections. More importantly, the end-to-end delay of

Communications & Multimedia Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (email: jau@csie.ntu.edu.tw)

a connection is reduced from $2H \cdot T$, as provided by stop-and-go, to $H \cdot T$, where H is the number of links (hops) the connection will pass by, and T is the interval of one time frame.

This paper is organized as follows. The next section contains the basic structure of the continuous framing mechanism. However, the statistical multiplexing gain is not provided in this basic form. An enhanced mechanism, namely *delay sending*, is then presented to provide intra-connection multiplexing gain. Another enhanced mechanism, *virtual tag*, is also described to provide inter-connection multiplexing gain.

CONTINUOUS FRAMING

Continuous framing consists of two components: an admission and congestion control policy imposed at the *source* node of each connection, and a service discipline at each *intermediate* node along the path. We use the concept of time framing as the central role of the service discipline, as in stop-and-go. The structure of the time framing strategy is illustrated in *Figure 1*, where packets arriving in the kth frame should be sent out in the $(k+1)$st frame. The basic concept of continuous framing is similar to that of the stop-and-go policy; hence, we will first briefly review the stop-and-go policy.

In stop-and-go, admission control and service discipline are based on each time frame, i.e. before a connection is set up, it has to reserve a certain bandwidth in each time frame. If the request is granted, the connection can be set up and a loss free transmission is guaranteed if the reservation is based on the peak rate of the connection. To provide a bounded end-to-end delay, a mechanism in all nodes should be implemented as below.

At each node, all packets arriving in one time frame from the incoming link should be sent out in the following time frame on its outgoing link. Therefore, if the length of one time frame equals T, the delay incurred per intermediate node has a value between 0 and T. However, since the time framing structures of the incoming and outgoing links are normally not synchronized, as shown in *Figure 2*, an extra delay is required for synchronization. This extra delay per node can be as high as T, hence the delay incurred per node has a value between 0 and $2T$. Thus, if there are H hops along the path from the source to the destination, the end-to-end delay can be bounded by $2H \cdot T$.

General concept of continuous framing

The objective of continuous framing is to remove the synchronization delay, as in the stop-and-go policy. To achieve this, the time framing structures of the incoming and outgoing links should be perfectly synchronized, i.e. for a connection j which passes through an intermediate node i with incoming link L_{in} and outgoing link L_{out}, node i should start a time frame on L_{out} for connection j as soon as a time frame finishes on L_{in}, as shown in

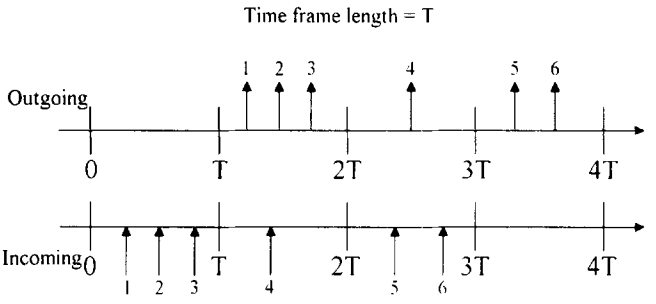

Figure 1 Structure of time framing strategy

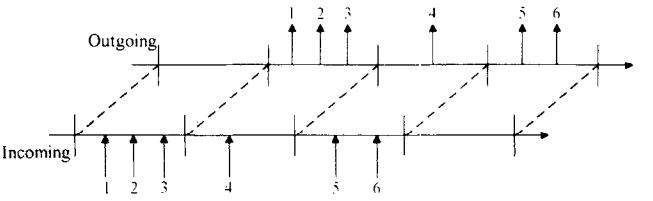

Figure 2 Structure of non-synchronized framing in stop-and-go

Figure 1. This perfect match of the time framings on incoming and outgoing links is the reason why this control mechanism is called continuous framing.

Since the framing structures on L_{in} and L_{out} are perfectly synchronized, there will be no synchronization delay at each node. Under such a mechanism, a packet will suffer only a delay of T on each intermediate node, hence the end-to-end delay can be bounded by $H \cdot T$. However, there are two problems to be considered for this mechanism:

1. For the output link L_{out}, there may be several connections passing through it, and their time frames may have different sizes and may not start or end at the same time. How can connection j be guaranteed that all packets in one frame from L_{in} can be sent out within the following frame from L_{out}?
2. How can the intermediate node know that a time frame has just finished on L_{in} for connection j such that it should start a new time frame on L_{out} for that connection?

To answer the first problem, we require the packets sent in one time frame to be evenly spread over the entire frame with the last packet sent at or near the end of the frame. For instance, if there are M packets to be sent in a frame with length T, then these packets should have an interval of length T/M between every two packets as shown in *Figure 3*. In this figure, the time intervals between packets (1, 2) and (2, 3) are $T/3$ and those between packets (3, 4), (4, 5), (5, 6) and (6, 7) are $T/4$. With such a constraint, the traffic pattern contributed by each connection would be uniform such that we can describe the data rate (or load) of the connection to be M/T packets/s without specifying the traffic pattern.

The advantage of doing this is to remove packet clustering of every connection. Hence, as long as the local load of all connections on L_{out} is smaller than unity, all packets in one incoming frame are guaranteed to be served in one frame on the outgoing link.

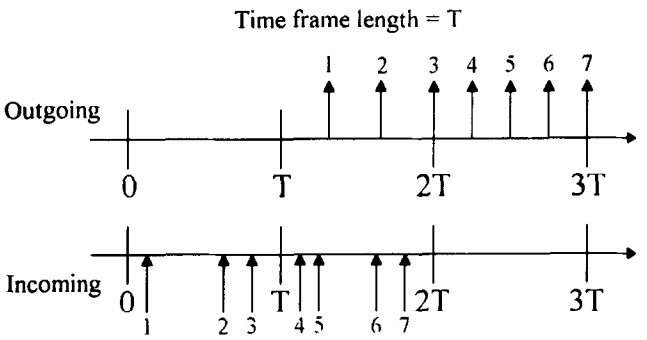

Figure 3 Sending packets uniformly within a frame

To answer the second problem, we simply add an *ending flag* on the last packet in a time frame for identification. Thus, as long as the intermediate node sees a packet with this flag, it knows that the incoming frame has been finished and a new frame should be started immediately on the outgoing link. An example is shown in *Figure 4*, where a dashed line represents the packet with the ending flag. With this mechanism, the time framings of different connections on a link do not have to begin or end at the same time.

In the following, we use the ATM network as an example to illustrate the concept, with the following assumptions:

1. The output link capacity can send 400,000 cells per second, equivalent to a bandwidth of 169.6 Mbit/s (=400,000*53*8).
2. There are in total six connections multiplexed on the output link, assuming the time frame is 10 ms for each connection. Hence, 4000 cells can be sent in one time frame on the output link. For each of explanation, we assume there are 4000 slots in one time frame, where one slot can be used to send one ATM cell.
3. Each of connections 1 and 2 will send 400 cells per frame, equivalent to a bandwidth requirement of 16.96 Mbit/s per connection. Each of connections 3 and 4 will send 200 cells per frame, equivalent to a bandwidth of 8.48 Mbit/s per connection. Lastly, each of connections 5 and 6 will send 40 cells per frame, equivalent to a bandwidth of 1.696 Mbit/s.

Based on the above conditions, and assuming that all cells have arrived in the previous time frame, connection 1 will use slots 10, 20, 30, ..., 3990 and 4000 to send its packets in the current time frame. The packet sent in slot 4000 should contain the ending flag. Similarly, connection 2 will use slots 9, 19, 29, ..., 3989 and 3999 for transmission; connection 3 will use slots 18, 38, 58, ..., 3978 and 3998 for transmission; connection 4 will use slots 17, 37, 57, ..., 3977 and 3997 for transmission; connection 5 will use slots 96, 196, 296, ..., 3896 and 3996 for transmission; finally, connection 6 will use slots 95, 195, 295, ..., 3895 and 3995 for transmission. Note that all packets in a connection are evenly spread out, and all packets are sent out within one time frame. This concept can be extended easily to connections requiring different frame sizes.

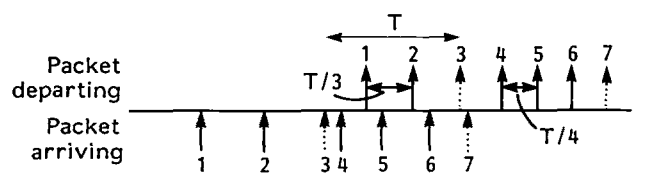

Figure 4 Time framing strategy with ending flag

Admission and congestion control mechanisms

Before further describing the protocol, we define the following notation:

- C: Capacity of the link (bits/s).
- L: Number of bits in a packet.
- T_i: Time frame length of the ith connection.
- M_i: Number of packets permitted to be sent during one time frame from the ith connection. It is also the bandwidth reserved by the ith connection.
- U_i: Utilization (load) of the ith connection on the link.
- $A_i(t)$: Number of packets actually arriving within time frame t from the ith connection.
- $D_i(t)$: Number of packets that can actually be sent in time frame $(t+1)$ from the ith connection.

In this section, we assume that all connections will make a bandwidth reservation based on their peak rates such that no packet loss may occur. Under this assumption, $A_i(t)$ will be equal to $D_i(t)$ and will never be greater than M_i.

The admission control mechanism is exercised before the connection is set up, to prevent the network from being overloaded. For connection i with the chosen T_i and M_i, its contributed load on each link will be $U_i = M_i \cdot L/T_i \cdot C$. Hence, for connection i to be admitted into the network, the following test should be exercised on each link it passes:

$$\sum_{i=1}^{n} U_i < 1 \qquad (1)$$

where there are in total n connections, including connection i, on this link. The test guarantees that all links along the path are not saturated.

The congestion control at all nodes can be implemented easily as follows. After receiving all packets within one time frame from the incoming link, say A_i, by detecting the ending flag a node will immediately start a new frame on the outgoing link and send these packets evenly with a gap T_i/A_i in between, as described earlier. With this control mechanism, an end-to-end delay bound and loss free transmission is ensured.

1. Continuous framing does not require extra delay for node synchronization as in stop-and-go, such that the end-to-end delay can be bounded by $H \cdot T$. Moreover, the jitter of continuous framing will be smaller than that of stop-and-go by removing the node synchronization delay.

2. For each switching node, the output traffic pattern of one connection will be reconstructed to be similar to its previous node[6,16]. Hence, a similar output traffic pattern can be maintained throughout the network to reduce the phenomenon of packet clumping and dispersion in the network[6,17].
3. Each connection is allowed to have an arbitrary time frame length, such that they do not have to be the multiple times of others, as required by stop-and-go.

Implementation issues

A problem of continuous framing is the computational complexity imposed on each intermediate node, since each connection requires the node to detect the beginning and end of the time frame, to compute the total number of packets received in the time frame, and to compute the spacing between packets on the outgoing link. To alleviate the loading of these intermediate nodes, an implementation method is proposed.

The idea is to move most of the computation requirements from intermediate nodes to source nodes, that is, the source node has to calculate the number of packets to be sent within each time frame and the time spacing between packets. This value of time spacing (timestamp) between packets should be put into the header of the previous packet. For instance, if packets i and $i+1$ should have a spacing of l, then a timestamp of l should be put in the header of packet i such that, whenever an intermediate node receives packet i, it can immediately pre-schedule a slot l time later for packet $i+1$ which is yet to come. This mechanism relieves the computational requirement of intermediate nodes.

Let us use *Figure 3* as an example, which represents the packet arrivals and departures at a source node. At time T, the source node realizes that there are three packets to be sent in the following frame; it will then put a timestamp of $T/3$ in packets 1 and 2. At time $2T$, the source node realizes there are four packets to be sent in the following frame, hence it puts a timestamp of $T/4$ in packets 3, 4, 5 and 6 and so on. With such a design, by reading the timestamp from a packet, the intermediate node can immediately schedule the transmission time of its following packet without further waiting and computation.

CONTINUOUS FRAMING WITH DELAY SENDING

In this section, an enhanced mechanism called *delay sending* is proposed to be used with continuous framing. The purpose of delay sending is to reduce the bandwidth requirement as compared to that of pure continuous framing, while retaining the same end-to-end delay and loss of free transmission.

Basic concept

Before discussion, we define a stream of packets to be (r,T)-*smooth* as that during any interval of length T, the total arrived packets have no more than $r \cdot T$ bits, where r stands for an upper bound of the traffic rate. In most cases, for a connection to satisfy both (r_1,T_1)-smooth and (r_2,T_2)-smooth, then $r_1 \geq r_2$ if $T_1 \leq T_2$, that is, given a larger time interval, there is a better chance to smooth out the burstiness of the traffic such that a smaller bandwidth is required. Thus, we may increase the time frame interval to reduce the bandwidth requirement.

However, if we simply increase the time frame size from T to T', the end-to-end delay will also be increased proportionally from $H \cdot T$ to $H \cdot T'$, so to reduce the peak rate demand for a connection without increasing its end-to-end delay from $H \cdot T$ to $H \cdot T'$, a modification of the above concept is required.

In pure continuous framing, all arriving packets within one time frame are guaranteed to be sent out in the following time frame, either at the source or at all intermediate nodes. In delay sending, the rule is slightly modified. At the *source* node, all arriving packets within one time frame would be guaranteed to be sent out in the following *two* frames. As shown in *Figure 5*, the connection has a bandwidth of three packets per time frame, and if four packets arrive in one frame, then packet number 4 is guaranteed to be sent out in the second following frame. Clearly, the peak rate assignment at the source node is done based on an interval of two time frames.

In other words, in pure continuous framing, the bandwidth requirement r_1, has to satisfy the (r_1,T)-smooth property, while the bandwidth requirement for delay sending, r_2, only has to satisfy the $(r_2,2T)$-smooth property. As mentioned earlier, r_2 is normally smaller than r_1, thus the bandwidth requirement can be reduced.

For all intermediate nodes, the rule remains the same as in pure continuous framing, i.e. all packets arriving in one time frame from the input link should be sent out in the following frame on the outgoing link. In the following, we examine the end-to-end delay incurred by delay sending. Since the packets may suffer a delay of up to $2T$ at the source node and one T for each intermediate node, the end-to-end delay can be bounded by $(H+1) \cdot T$. Therefore, a bandwidth saving can be achieved at the cost of an increased end-to-end delay from $H \cdot T$ to $(H+1) \cdot T$. If H is large, the portion of increased delay will be insignificant.

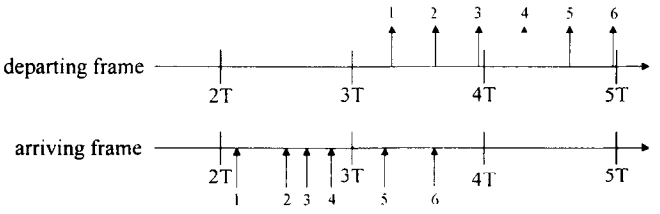

Figure 5 Service discipline of continuous framing with delay sending

Additionally, if we allow packets to be sent out within the following three frames at the source node, then the connection only requires a transmission rate r_3 to satisfy the $(r_3,3T)$-smooth property. Similarly, the end-to-end delay will become $(H+2)\cdot T$ in this case. Likewise, we can find r_4 to satisfy the $(r_4,4T)$-smooth property, etc. Again, the magnitudes of these transmission rates will normally have the relationship $r_1 \geqslant r_2 \geqslant r_3 \geqslant r_4 \geqslant \ldots$, hence more bandwidth can be saved. However, the end-to-end delay will be increased by T, $2T$, $3T$ or $4T$, respectively. Basically, we can decrease the bandwidth requirement by delaying more time frames at the source node at the cost of an increased delay. Hence, choosing a proper number of delayed frames will depend upon the performance requirement of connections.

In fact, we can reduce the bandwidth requirement while keeping the same end-to-end delay. To make the end-to-end delay of delay sending (with one extra frame) the same as that of the pure continuous framing method, the following condition should be satisfied:

$$H\cdot T = (H+1)\cdot T' \tag{2}$$

where T' is the time frame interval of the pure continuous framing and T' is that of delay sending. Hence, we have:

$$T' = \frac{H}{H+1}\cdot T \tag{3}$$

For example, if $T = 10$ and $H = 9$, then $T' = 9$. In this case, if we denote the bandwidth requirement using pure continuous framing as r_1 and that using delay sending as r_2, then r_1 has to satisfy $(r_2, 18)$-smooth. Thus, r_2 will normally be smaller than r_1, and hence a bandwidth saving is obtained. However, there are a few cases with special traffic patterns such that $r_1 < r_2$, then we simply use the pure continuous framing mechanism. That is, an alternative is provided by considering delay sending together with pure continuous framing to save bandwidth. Thus, the bandwidth requirement for the connection may be reduced while the end-to-end delay remains the same.

Examples

In this subsection, we present two examples to demonstrate the effect of bandwidth saving by using delay sending. The first example uses a probabilistic model to simulate the packet arrival process. This model assumes that there are T slots in one time frame, where each slot can be used to transmit one packet. We also assume the average arrivals of a connection to be N packets per time frame. Therefore, we model the arrival process as a Bernoulli process with packet arrival rate $p = N/T$ per time slot for this connection.

Based on this model, the number of packet arrivals during one time frame is a random variable, ranging from 0 to T, with an average of N. Hence, to guarantee a loss free transmission, the bandwidth reservation will be T slots per time frame, which will use up all the channel bandwidth, and will be impractical. Hence, in this example, the connection will not require a loss free transmission. Instead, the connection will reserve a bandwidth to guarantee a packet loss probability to be smaller than a given P_{loss}.

We define K as the number of reserved slots per time frame for the connection. Then, given T, p and K, we have:

$$\frac{\begin{bmatrix} T \\ K+1 \end{bmatrix} p^{K+1}(1-p)^{T-K-1} + \begin{bmatrix} T \\ K+2 \end{bmatrix} p^{K+2}(1-p)^{T-K-2}\cdot 2 + \cdots + \begin{bmatrix} T \\ T \end{bmatrix} p^{T}\cdot(T-K)}{T\cdot p} < P_{\text{loss}}$$

By simplifying it, we have:

$$\sum_{K+1 \leqslant i \leqslant T} \begin{bmatrix} T \\ i \end{bmatrix} p^i(1-p)^{T-i}(i-K) < T\cdot p\cdot P_{\text{loss}}$$

From this equation, we can compute K from the given P_{loss}. In the following, we find and compare the values of K for pure continuous framing and delay sending under the condition that they have the same end-to-end delay. *Figure 6* shows the results of bandwidth requirements for pure continuous framing and delay sending with one extra frame delayed at the source node. Note that in this figure, the bandwidth requirements are normalized with respect to T. The parameters used in *Figure 6* are $T = 20$, $H = 9$ and $P_{\text{loss}} = 10^{-4}$. We can observe that the bandwidth requirement for delay sending is always smaller than that of pure continuous framing, while the end-to-end delay remains the same. Moreover, for $p < 0.7$, the average bandwidth saving for delay sending is about 25%, which is very significant. *Figure 7* shows the results by changing T from 20 to 30, and a similar conclusion is obtained.

Figure 8 gives another result, which shows the effect of delaying more frames at the source node with $p = 0.1$, $T = 10$, $H = 9$ and $P_{\text{loss}} = 10^{-4}$. Again, the end-to-end delay remains the same for all cases. Clearly, more bandwidth saving can be achieved if packets can be delayed for more time frames at the source node.

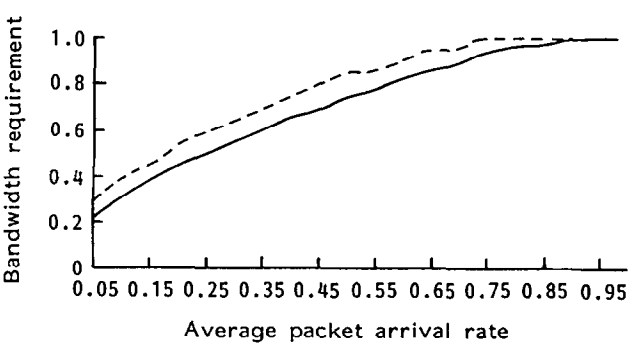

Figure 6 Comparison of bandwidth requirement with and without delay sending ($T = 20$, $H = 9$, $P_{\text{loss}} = 10^{-4}$)

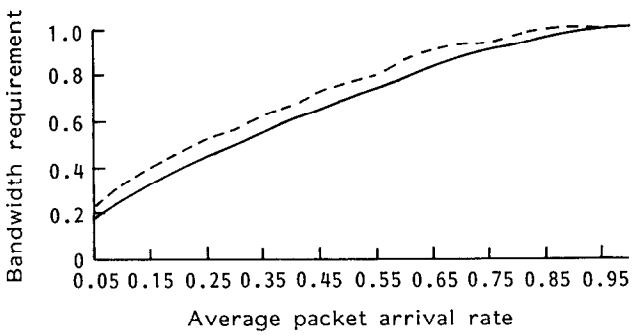

Figure 7 Comparison of bandwidth requirement with and without delay sending ($T = 30$, $H = 9$, $P_{loss} = 10^{-4}$)

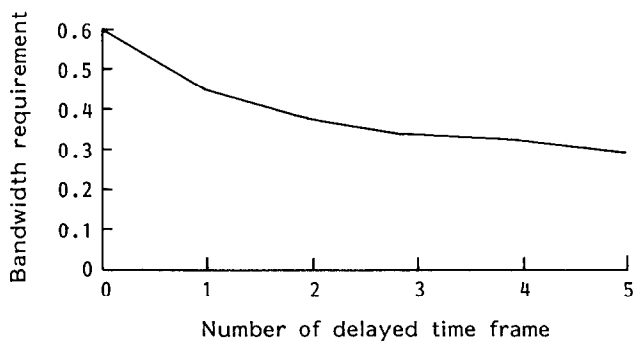

Figure 8 Comparison of bandwidth requirement by varying number of delayed frames ($T = 10$, $H = 9$, $p = 0.1$, $P_{loss} = 10^{-4}$)

For the second example, we use a train model as the packet arrival process. The packet arrival process can be described as an ON-OFF-ON-OFF model. Similar to the previous example, we divide time into slots where one slot can be used to transmit one packet. When the arrival process is in the ON state, one packet will arrive in each slot. While the process is in the OFF state, no packets will arrive. This model can be described by a 4-tuple (N_1, F_1, N_2, F_2) as that within one cycle, there are N_1 consecutive packet arrivals, followed by F_1 idle slots, which are followed by another N_2 consecutive packet arrivals and another F_2 idle slots. This pattern will repeat itself to constitute the arrival process. This model can be viewed as a second degree ON-OFF train model to obtain a more bursty traffic pattern.

For this type of arrival process, the bandwidth reservation can be made to achieve a loss free transmission. We can easily derive the bandwidth requirement of connections, as shown in *Figure 9*, by varying the value of N_1 with $N_2 = 3$, $F_1 = F_2 = 6$, $H = 9$ and $T = 10$. Similarly, the end-to-end delay for cases with or without delay sending are set equal. This figure shows that the bandwidth saving ranges from 12% ($N_1 = 5$) to 40% ($N_1 = 10$, 11, 1nd 12). This result again shows the efficiency in bandwidth saving by using delay sending. Moreover, the more bursty the packet steam is, the more bandwidth saving there will be. The results of another set of parameters ($T = 20$, $N_1 = 3N_2$, $F_1 = F_2 = 6$, $H = 9$) are illustrated in *Figure 10*, and a similar behaviour is also observed.

If we think carefully about delay sending, it is essentially a mechanism of *intra*-connection multiplexing between time frames of the same connection. That is, if there are more packets than the connection is granted in one time frame, these extra packets will probably find unused bandwidth in the following time frame for transmission. With this concept in mind, we will naturally think of *inter*-connection multiplexing, where one connection with extra packets can use unused bandwidth from other connections. This will be described in the following section.

CONTINUOUS FRAMING WITH VIRTUAL TAG

For many multimedia applications, the traffic is so bursty that it is impractical to reserve the bandwidth based on its peak rate. MPEG encoded video stream offers a good example in which an I-frame generates a much larger amount of data than a B-frame. Hence, it is more practical to reserve the bandwidth to be smaller than its peak rate. However, whenever the input data of a connection exceeds its reserved bandwidth in some time frames, it would be desirable that it can use the unused bandwidth of other connections, if there is any.

In this service mechanism, each connection sends as many packets as it can up to its reservation. Moreover, it also monitors the bandwidth usage of other connections in order to use their unused bandwidth. This concept is very similar to that of leaky bucket with

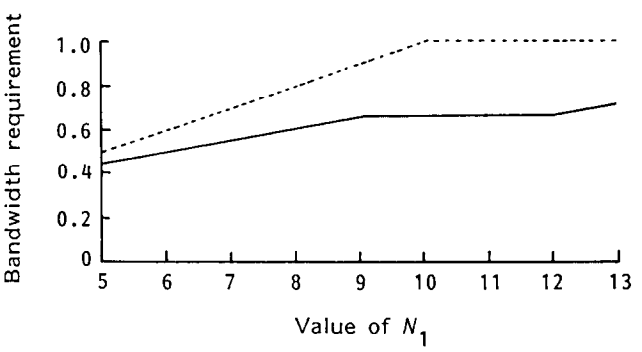

Figure 9 Comparison of bandwidth requirement using train model ($T = 10$, $N_2 = 3$, $F_1 = F_2 = 6$, $H = 9$). ·····: Without delay sending; ———: with delay sending.

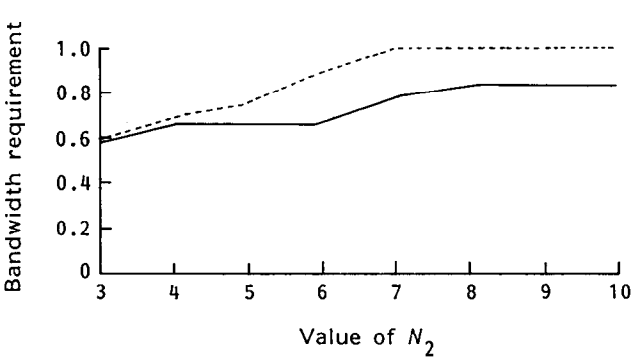

Figure 10 Comparison of bandwidth requirement using train model ($T = 20$, $N_1 = 3N_2$, $F_1 = F_2 = 6$, $H = 9$). ·····: Without delay sending; ———: with delay sending.

virtual tag. The service operations of this mechanism are similar to those of pure continuous framing, with the following modification.

At the beginning of one time frame, connections are assigned into two groups, groups U and V, according to the number of packet arrivals within the previous time frame. By using the notation defined earlier, a connection will belong to group U if $A_i \leqslant M_i$, i.e. the number of arriving packets does not exceed its reservation. Otherwise, this connection will belong to group V. For group U connections, the service discipline is the same as in pure continuous framing.

For a group V connection $i(A_i > M_i)$, we first allocate M_i slots to this connection within the current time frame, and these slots are scattered evenly with a gap T_i/M_i in between, as usual. Then, of all A_i packets, M_i of those with a higher priority will definitely be sent out in the current time frame using the reserved bandwidth. For the remaining $A_i - M_i$ packets, which are packets with a lower priority, they will be tagged and the number of tagged packets will be recorded in a counter. The channel server will examine all connections to see if there is any empty slot unused by Group U connections. If there is any, it can be used to transmit a tagged packet. If, at any following intermediate node, idle slots cannot be found for these tagged packets, they will be dropped.

CONCLUSIONS

A novel congestion control strategy, continuous framing, for delay sensitive traffic in broadband networks is presented. Two enhanced mechanisms, delay sending and virtual tag, are also proposed to increase the channel efficiency. Moreover, these two enhanced mechanisms can be combined together for use with continuous framing simultaneously, since one provides intra-connection multiplexing and the other provides inter-connection multiplexing, and there is no conflict between them in implementation. Hence, the channel utilization can be further increased. In summary, the main contributions of this paper are:

1. Continuous framing provides bounded end-to-end delay and loss free transmission, as in stop-and-go. Moreover, the phenomenon of packet clustering is removed.
2. The bounded end-to-end delay caused by using continuous framing is reduced from $2H \cdot T$, as in stop-and-go, to $H \cdot T$. This is a significant improvement.
3. Continuous framing allows frames to be of arbitrary lengths to satisfy various traffic patterns and real-time constraints. This is difficult to achieve with stop-and-go.
4. By using continuous framing with delay sending, bandwidth saving can be achieved without sacrificing performance quality, such as end-to-end delay.
5. By using continuous framing with virtual tag, inter-connection multiplexing can be obtained to improve channel efficiency, and the packets are still served in sequence. Moreover, it provides firewalls among connections, such that even if a connection sends more packets than its reservation, other connections will not be affected.
6. By using continuous framing with delay sending and virtual tag, both intra- and inter-connection multiplexing gains can be achieved. Hence, the bandwidth utilization can be further increased without losing other good properties.

REFERENCES

1 Bae, J J and Suda, T 'Survey of traffic control schemes and protocols in ATM networks', *Proc. IEEE* (February 1991) pp 170–189

2 Hong, D, Suda, T and Bae, J J 'Survey of techniques for prevention and control of congestion in an ATM network', *Proc. IEEE ICC*, Denver, CO (1991) pp 204–210

3 Jacobson, V 'Congestion avoidance and control', *Proc. ACM SIGCOMM*, Stanford, CA (1988) pp 314–329

4 Yin, N and Hluchyj, M G 'A dynamic rate control mechanism for integrated networks', *Proc. IEEE INFOCOM*, Bal Harbor, FL (1991) pp 543–552

5 Zhang, H and Keshav, S 'Comparison of rate-based service disciplines', *Proc. ACM SIGCOMM*, Zurich, Switzerland (1991) pp 113–121

6 Zhang, H and Ferrari, D 'Rate-controlled static-priority queueing', *Proc. IEEE INFOCOM*, San Francisco, CA (1993) pp 227–236

7 Zhang, L 'VirtualClock: A new traffic control algorithm for packet switching networks', *ACM Comput. Comm. Rev.*, Vol 20 No 4 (1990) pp 19–29

8 Golestani, S J 'Congestion-free transmission of real-time traffic in packet networks', *Proc. IEEE INFOCOM*, San Francisco, CA (1990) pp 527–542

9 Bolestani, S J 'A stop-and-go queueing framework for congestion management', *Proc. ACM SIGCOMM*, Philadelphia, PA (1990) pp 8–18

10 Golestani, S J 'Duration-limited statistical multiplexing of delay-sensitive traffic in packet networks', *Proc. IEEE INFOCOM*, Bal Harbor, FL (1991) pp 323–332

11 Golestani, S J 'Congestion-free communication in high-speed packet networks', *IEEE Trans. Comm.*, Vol 39 No 12 (December 1991) pp 1802–1812

12 Kalmanek, C R and Kanakia, H 'Rate controlled servers for very high-speed networks', *Proc. IEEE GLOBECOM*, San Diego, CA (1990) pp 12–20

13 Demers, A, Keshav, S and Shenker, S 'Analysis and simulation of a fair queueing algorithm', *Proc. ACM SIGCOMM*, Austin TX (1989) pp 1–12

14 Ferrari, D and Verma, D 'A scheme for real-time channel establishment in wide-area networks', *IEEE J. Selected Areas in Comm.*, Vol 8 No 3 (1990) pp 368–379

15 Verma, D, Zhang, H and Ferrari, D 'Guaranteeing delay jitter bounds in packet switching networks', *Proc. Tricomm*, Chapel Hill, NC (April 1991) pp 35–46

16 Verma, D and Ferrari, D 'Variation of traffic parameters in ATM networks', *Proc. IEEE ICC'*, Chicago, IL (1992) pp 689–693

17 Guillemin, F and Monin, W, 'Management of cell delay variation in ATM networks', *Proc. IEEE GLOBECOM*, Orlando, FL (1992) pp 128–132