A LOSS-FREE CONNECTION CONTROL PROTOCOL FOR THE THUNDER AND LIGHTNING NETWORK

Emmanouel (Manos) Varvarigos   Vishal Sharma

Department of Electrical and Computer Engineering
University of California
Santa Barbara, CA 93106-9560

Abstract— The *Ready-to-Go Virtual Circuit* protocol (or RGVC) is a novel connection control protocol for gigabit networks that is designed to ensure lossless transmission for delay-sensitive traffic and for traffic whose rate changes with time. The RGVC protocol is one of the two connection control protocols that will be used in the 40 Gbit/s fiber-optic ATM-based Thunder and Lightning network, currently being developed at UCSB. In this paper, we introduce the RGVC protocol, discuss its main features, and indicate its lossless character. The RGVC protocol can be viewed as a reservation protocol where the reservation and the data transmission phases overlap. The source need not wait for an end-to-end round-trip delay for reservations to be made before transmitting the data. Instead the data packets follow the setup packet after a short *offset-interval*, which is much smaller than the round-trip delay. As a result, the protocol does considerably better than wait-for reservation protocols in terms of minimizing pre-transmission delay, and is useful for connection establishment for traffic with strict delay requirements. If the setup packet is unsuccessful in reserving the required capacity, or if the rate of the session changes without there being sufficient capacity to accommodate the change, the packets are buffered at intermediate nodes and back-pressure is exercised to the upstream nodes to control the source transmission rate. The back-pressure mechanism uses the concept of *freezing* of capacity to ensure that the protocol is lossless.

I. Introduction

The Thunder and Lightning network [1] is a very high-speed fiber-optic communications network being designed and built at UCSB, which is projected to carry both constant-rate traffic and delay-sensitive, variable-rate traffic. Our objectives in designing the connection and flow control algorithms for this network are to ensure lossless transmission, efficient utilization of capacity, minimum pre-transmission delay for delay-sensitive traffic, and packet arrival in correct order. To meet these objectives, we have proposed two new connection control protocols: the Efficient Reservation Virtual Circuit protocol (or ERVC) that will be used for constant-rate sessions, or for sessions whose rate has some particular smoothness properties, and the Ready-to-Go Virtual Circuit protocol (or RGVC) that will be used for delay-sensitive traffic or for traffic whose rate changes slowly with time. The ERVC protocol, described in [2], uses reservations and requires little buffering for constant-rate sessions. The RGVC protocol, which is the subject of this paper, uses back-pressure and requires buffering at intermediate nodes.

If a session is critically delay-sensitive and cannot tolerate the round-trip propagation delay required for call setup by the ERVC protocol, the RGVC protocol is employed to establish the connection. In the RGVC protocol, a setup packet is first transmitted over a path towards the destination, followed after a short *offset-interval* by the data packets (see Fig. 1). Therefore, in the RGVC protocol, a pipelining between the setup phase and the data-transmission phase is achieved, reducing the pre-transmission delay to the minimum possible. This differs from wait-for reservation virtual circuit (henceforth called WRVC) protocols, where a pre-transmission delay at least equal to one round-trip propagation delay is needed before the data transmission phase can begin (this can be as large as 30 ms for coast-to-coast communication). If the setup packet is successful in reserving the required capacity and the session rate remains constant, the RGVC protocol resembles a usual reservation protocol, with the added advantage that the capacity is blocked for other sessions for a much smaller time than in WRVC protocols, because capacity is reserved only for the duration of the session plus the duration of the short offset-interval. If the residual capacity of a link on the path is less than the requested rate, or if the rate of the session changes with time, the session is granted the maximum permissible rate and back-pressure (the details of which we provide in Sections 4 and 5) is exercised to control the source transmission rate. Back-pressure is exercised by buffering the excess packets at the intermediate node and transmitting to the previous nodes a control packet that causes them to take appropriate action to control their their rate.

The RGVC protocol can operate either with RAM buffers or with FIFO buffers at the network nodes. With RAM buffers, it is possible to throttle a particular session without affecting other sessions sharing the same buffer, because a separate logical queue can be maintained for each session. That is, per session queueing can be exercised. With FIFO buffers, however, it is not possible to throttle an individual session, because its packets can

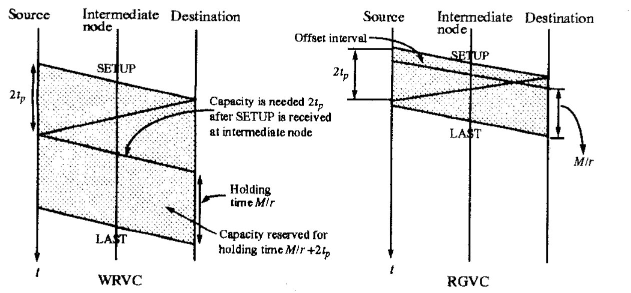


Fig. 1. Illustrates the advantage of the RGVC protocol. In the WRVC protocol, the capacity is blocked for duration equal to $\frac{M}{r} + 2t_p$, where t_p is the propagation delay between the source and destination. In addition, the session cannot begin transmission till $2t_p$ after the call setup procedure is initiated. In the RGVC protocol, the session can begin transmission immediately following the *offset-interval*, and capacity is only occupied for time $\frac{M}{r}$ plus the duration of the offset interval. The above illustrations correspond to the case where the setup packet is successful in making the appropriate reservations.

no longer be isolated from packets of the other sessions that share the same buffer. Thus, the actions that a node takes upon the receipt of a throttle packet differ depending on the nature of the buffers, RAM or FIFO, that are used at the nodes, and are discussed in detail in Sections 4 and 5, respectively. Whereas RAM buffers provide for simple operation at the expense of buffer management, FIFO buffers provide for simple buffer management at the expense of a more complex back-pressure mechanism. This, however, is the price we pay for a much simpler FIFO buffer implementation in very high-speed networks, such as the Thunder and Lightning network, where operational speeds of tens of gigabits per second render RAM buffers infeasible.

Since the advent of high-speed integrated networks in the mid 1980's, high-speed networks of different types have been designed, built, and studied by a number of researchers in the United States (see for example [3], [4], [5], and [6]), Japan, and Europe (see for example [7], [8], [9], [10], and [11]). The important issues of call establishment, connection control, and bandwidth management in such networks have also been an active area of research within the community (see for instance [12], [13], [14], [15], [16], [17], and [18]). Partridge [19] presents an interesting discussion of the goals and challenges of gigabit networking, which is rapidly becoming a reality with the recent advances in VLSI technology and fiber-optic transmission systems.

The remainder of the paper is organized as follows. In Section 2 we provide a general description of the RGVC protocol, and discuss briefly the role of the main control packets used. In Section 3, we explain the switch architecture that we assume. In Section 4 we describe the operation of the protocol with a RAM buffer implementation. We discuss the difficulties posed by FIFO buffers, and explain the operation of the protocol with a FIFO buffer implementation in Section 5. Concluding remarks follow in Section 6.

II. General description of the RGVC protocol

In our description of the RGVC protocol, we do not consider issues related to error control and retransmission, since in the Thunder and Lightning network these functions are performed at the transport layer. In the RGVC protocol, a SETUP packet is transmitted first over the path to reserve the required capacity and set the routing tables, and is followed after the offset-interval by the data packets. Once a setup packet is processed at a switch and makes the needed reservations, the data packets can be routed through the switch without any (or with minimal) processing overhead. The offset-interval is equal to the number of hops on the path times the difference in the processing times of a setup packet and a data packet at a node. Therefore, the offset-interval is the minimum time by which the connection setup phase and the data transmission phase must be separated to ensure that the data packets do not overtake the setup packet.

When the residual capacity at an intermediate node is sufficient to accommodate the session, the node reserves capacity for that session and forwards the SETUP packet along the path. Otherwise, the node reserves the maximum available capacity for the session, reduces the requested rate field to the rate granted to the session, and forwards the the SETUP packet to the next node. The node also transmits to the previous node on the path a control packet, which informs it that the entire capacity requested by the session could not be allocated to it, and requests appropriate actions to control the transmission rate of the session. The previous node *freezes* capacity corresponding to the buffer space taken at the next node, which means that this capacity is not available for use by sessions until this buffer space becomes free. Therefore, a key concept in the RGVC protocol is that the capacity available on a link is coupled with the free buffer space at the next node in order to ensure that no buffer overflow occurs. The way this is done is described in Section 4 (for RAM buffers) and Section 5 (for FIFO buffers), respectively. A node *defreezes* all or part of this capacity only when, based on its estimates, the buffer space occupied at the next node decreases. Each node has for every incoming link, buffer space equal to at least $2t_pC$, where t_p is the propagation delay on that link and C is the link capacity. The exact buffer requirements depend on whether the FIFO or the RAM implementation of the protocol is being used, and on the trade-offs desired between protocol implementation complexity and the buffer space per node.

The LAST packet is transmitted by the source after all data packets of the session have been transmitted. It signals that the session has terminated, which allows the intermediate nodes to release the reserved capacity and update the routing tables.

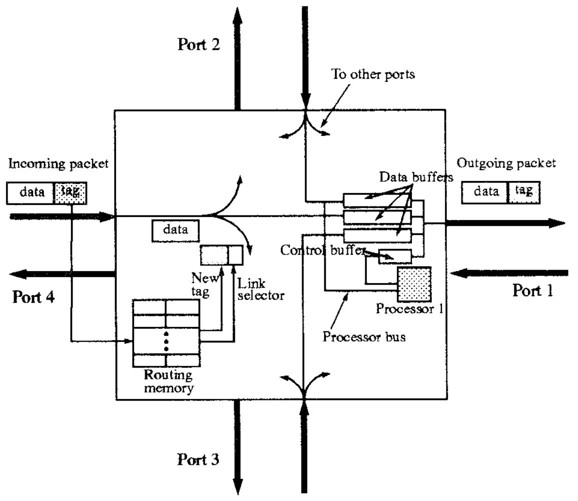


Fig. 2. Illustrates the architecture of Thunder and Lightning switch, for which $k = 4$. The routing tag (or header) of an incoming packet addresses the routing memory (set during the connection setup phase), which outputs the new routing tag and the link selector. In the link selector, a bit is set for each outgoing link for which the packet is intended (required, for example, in multicast operations), and the packet is accepted only by the corresponding data buffer(s). We show the details only for Port 1.

III. Switch architecture

In this section, we describe the switch architecture that we will refer to in our description of the protocol in the subsequent sections.

A network switch has k bidirectional ports, each of which corresponds to an incoming and an outgoing link. Each port has a processor, called switch port processor (or SPP), which is responsible for processing the control packets flowing through the outgoing link of that port (see Fig. 2). An outgoing link transmits packets from k buffers, which can either be RAM buffers or FIFO buffers. Of these buffers, $k-1$ buffers, called *data buffers*, are used by the packets arriving on the other incoming links and intended for transmission via this link, while the k^{th} buffer, called *control buffer*, is used by control packets. The switching hardware handles the movement of the data packets through the switch without involving the processor. The control buffer has priority over the data buffers, so that the control packets are transmitted without being affected by the data packets. If a node is a source for a session using a port j, one of the data buffers of port j is connected to it. A data buffer n at node i is denoted by $Q_i(n)$. We use the notation $Q_i(\mathcal{S})$ for the buffer used by a session $\mathcal{S}$ at node i, and for the set of sessions that share that buffer, and we use the notation $|Q_i(n)|$ to denote the buffer space occupied at $Q_i(\mathcal{S})$.

In the 40 Gbit/s Thunder and Lightning network, each switch has $k = 4$ ports, and uses FIFO buffers. Although the use of FIFO buffers poses some difficulties in the design of the RGVC protocol that are not present in the RAM case, this is the price paid for the ease of a FIFO buffer hardware implementation at speeds of 40 Gbit/s at which the use of RAM buffers is no longer feasible.

IV. RGVC Protocol with RAM buffers

In the RAM case, we can control the rate of an individual session, and so we can describe the operation of the protocol for a particular session $\mathcal{S}$. The path followed by session $\mathcal{S}$ is denoted by $s_0, s_1, \ldots, s_h$, where s_0 is the source node and s_h is the destination node. To describe the operation of the RGVC protocol with RAM buffers, we will use the following notation for the variables associated with session $\mathcal{S}$.

$R_i(\mathcal{S})$ = allowable rate for session $\mathcal{S}$ at node s_i.
$B_i(\mathcal{S})$ = buffer space occupied by packets of session $\mathcal{S}$ at node s_i.
$A_i(\mathcal{S})$ = available capacity for session $\mathcal{S}$ at node s_i.
$F_i(\mathcal{S})$ = capacity *frozen* due to session $\mathcal{S}$ at node s_i.
$R^i_{i+1}(\mathcal{S})$ = allowable rate for session $\mathcal{S}$ at node s_{i+1}, as known at node s_i.
$B^i_{i+1}(\mathcal{S})$ = buffer space occupied by packets of session $\mathcal{S}$ at node s_{i+1}, as known at node s_i.
$R^i_{i-1}(\mathcal{S})$ = allowable rate for session $\mathcal{S}$ at node s_{i-1}, as known at node s_i.

When the rate of a session at a node s_i (including the source) changes, a RATE packet containing $R_i(\mathcal{S})$ is sent to the next node on the path. If the available capacity at node s_{i+1} is sufficient, $R^{i+1}_i(\mathcal{S})$ is updated to $R_i(\mathcal{S})$, and the node forwards the RATE packet. If the available capacity is insufficient, node s_{i+1} allocates (as described later) capacity $R_{i+1}(\mathcal{S})$ to session $\mathcal{S}$ and buffers the excess packets. At the same time, a THROTTLE packet containing the new rate $R_{i+1}(\mathcal{S})$ and the buffer space $B_{i+1}(\mathcal{S})$ occupied at node s_{i+1}, is sent to node s_i. Node s_i then updates the frozen capacity due to session $\mathcal{S}$ according to

$$F_i(\mathcal{S}) = \frac{B^i_{i+1}(\mathcal{S}) + D_i(\mathcal{S}) - 2t_{i,i+1}R^i_{i+1}(\mathcal{S})}{2t_{i,i+1}}, \quad (1)$$

where $D_i(\mathcal{S})$ is the amount of data that node s_i transmitted to node s_{i+1} in the $2t_{i,i+1}$ seconds prior to the arrival of the THROTTLE packet (we explain shortly how to obtain the estimates $D_i(\mathcal{S})$), and $2t_{i,i+1}$ is round-trip propagation delay between node s_i and node s_{i+1}. The estimate $B^i_{i+1}(\mathcal{S})$, of the buffer space at node s_{i+1} as known at node s_i, does not include $D_i(\mathcal{S})$ because this data had not arrived at node s_{i+1} when the THROTTLE packet was sent. Note that if $R^i_{i+1}(\mathcal{S})$ does not change (in which case $F_i(\mathcal{S})$ will be updated again) $B^i_{i+1}(\mathcal{S}) + D_i(\mathcal{S}) - 2t_{i,i+1}R^i_{i+1}(\mathcal{S})$ is the buffer space that will be occupied at node s_{i+1} due to session $\mathcal{S}$. The THROTTLE packet sent from node s_{i+1} to node s_i also has a field to record the cumulative buffer space $\sum_{j=i+1}^{h-1} 2t_{j,j+1}F_j(\mathcal{S})$ occupied at the nodes ahead, and is used by the source to calculate the time for which it should cease transmission (see Fig. 3).

The capacity $A_i(\mathcal{S})$ available for a session $\mathcal{S}$ at the outgoing link of node s_i is equal to

$$A_i(\mathcal{S}) = C - \sum_{\substack{\text{all } \mathcal{S}' \\ \mathcal{S}' \neq \mathcal{S}}} R_i(\mathcal{S}') - \sum_{\mathcal{S}' \in \mathcal{Q}_{i+1}(\mathcal{S})} F_i(\mathcal{S}'), \qquad (2)$$

where $\mathcal{Q}_{i+1}(\mathcal{S})$ is the set of sessions that use the same buffer at node s_{i+1} as session $\mathcal{S}$.

Observe that in the above equation the available capacity excludes frozen capacity, that is, the capacity that corresponds to the buffer space occupied at node s_{i+1} by packets of sessions that use the same buffer as session $\mathcal{S}$. Thus, the total permissible input rate of buffer $\mathcal{Q}_{i+1}(\mathcal{S})$ at a particular instant is always less than what it has space to store at that time without overflow. When the capacity available on its incoming link is equal to $A_i(\mathcal{S})$, buffer $\mathcal{Q}_{i+1}(\mathcal{S})$ has space to store $2t_{i,i+1}A_i(\mathcal{S})$ bits, which is adequate to store the (maximum of) $2t_{i,i+1}A_i(\mathcal{S})$ bits that node s_{i+1} may receive before the input rate of session $\mathcal{S}$ reduces and becomes equal to its output rate. This is because it takes time $2t_{i,i+1}$ for a THROTTLE packet sent by node s_{i+1} to travel to node s_i, for node s_i to reduce its rate to $R^i_{i+1}(\mathcal{S})$, and for the reduction in rate to become effective at node s_{i+1}. Thus, freezing of capacity ensures that the communication is lossless.

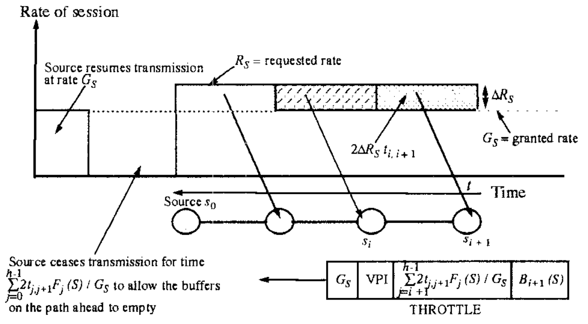
Finally, node s_i allocates the rate $R_i(\mathcal{S})$ to session $\mathcal{S}$ according to

$$R_i(\mathcal{S}) = \min(A_i(\mathcal{S}), R^i_{i-1}(\mathcal{S}) + \frac{B_i(\mathcal{S})}{T_p}), \qquad (3)$$

where T_p is a parameter, which controls the rate at which the buffer at node s_i should be emptied. The $R_i(\mathcal{S})$ allocated to session $\mathcal{S}$ is updated whenever $A_i(\mathcal{S})$ or $R_{i-1}(\mathcal{S})$ changes, or after time T_p, whichever comes first. The above choice of the rate ensures that the outgoing rate of session $\mathcal{S}$ at node s_i includes the incoming rate plus the rate at which the node should transmit to empty its buffer within time T_p.

When the source node receives the THROTTLE packet, it stops transmission until its estimate of the buffer space occupied by packets of this session at the nodes of the path becomes zero. That is, the source ceases transmission for a time equal to $\sum_{j=0}^{h-1} 2t_{i,i+1}F_j(\mathcal{S})/G_{\mathcal{S}}$, where $G_{\mathcal{S}}$ is the rate granted to the session at the node at which the THROTTLE packet originated (both quantities are contained in the THROTTLE packet), which allows the buffers at the nodes ahead to empty (see Fig. 3).

We now explain how a node s_i obtains an estimate of the amount of data $D_i(\mathcal{S})$ that it transmits to node s_{i+1} in an interval $2t_{i,i+1}$. Since the rate of a session $\mathcal{S}$ at a node s_i will be a stepwise function, conceptually the node can obtain this estimate simply by calculating the area under the rate-function (that is, the function which represents how the rate of a session changes with time) of session $\mathcal{S}$. In practice, an efficient way for the node to do this is to maintain a linked list, each of whose elements



Rate of session
Source resumes transmission at rate G_S
R_S = requested rate
ΔR_S
G_S = granted rate
$2\Delta R_S t_{i,i+1}$
t
Time
Source s_0
s_i
s_{i+1}
Source ceases transmission for time $\sum_{j=0}^{h-1} 2t_{j,j+1}F_j(S)/G_S$ to allow the buffers on the path ahead to empty
G_S
VPI
$\sum_{j=i+1}^{h-1} 2t_{j,j+1}F_j(S)/G_S$
$B_{i+1}(S)$
THROTTLE

Fig. 3. Illustrates the actions of the source node on receiving a THROTTLE packet. The x-axis represents time. In the above figure, the rate available at s_{i+1} at time t is equal to $G_{\mathcal{S}}$, while the rate at which packets arrive at s_{i+1} is $G_{\mathcal{S}} + \Delta R_{\mathcal{S}}$. Thus, at time t node s_{i+1} transmits a THROTTLE packet. At time $t + t_{i,i+1}$, node s_i receives the THROTTLE packet, reduces its rate to $G_{\mathcal{S}}$, and forwards the THROTTLE packet to node s_{i-1}. Finally, at time $t + \sum_{j=0}^{j=i} t_{j,j+1}$, the source receives the THROTTLE packet and ceases transmission for time $\sum_{j=0}^{h-1} 2t_{j,j+1}F_j(\mathcal{S})/G_{\mathcal{S}}$, which clears the buffer space occupied by session $\mathcal{S}$ at the nodes on the path.

stores the rate and the corresponding time at which the rate-function made jumps in the $2t_{i,i+1}$ seconds preceding the current time. In other words, the linked list can be thought of as a sliding window of length $2t_{i,i+1}$, which captures a snapshot of the profile of the rate-function in the last $2t_{i,i+1}$ seconds. The data transmitted to the subsequent node is then obtained by using these rates and times to calculate the area under the rate-function in the last $2t_{i,i+1}$ seconds. The details of this approach are given in [20].

V. RGVC protocol with FIFO buffers

We saw in Section 4 that when the nodes of the network use RAM buffers, an individual session can be throttled, thus allowing other sessions sharing the same outgoing link to continue operation at their assigned rates. However, when the nodes of the network use FIFO buffers, which is the case for the Thunder and Lightning network, this flexibility is lost and several difficulties arise. Since a node cannot isolate the packets of a particular session, control over the rate of an individual session is lost and the transmission rate of the entire FIFO through which the session is routed has to be reduced (as opposed to reducing only the rate of the session in question). As a result, other sessions sharing the FIFO are also temporarily affected.

In this section we discuss a scheme for the operation of the RGVC protocol with FIFO buffers, based on the concept of *buffer partitioning*. Buffer partitioning ensures that the number of successive THROTTLE packets transmitted by each node to its preceding node is bounded by a constant, and that small fluctuations in the rates of the sessions are smoothed out so that the network does not need to respond to every change in the rate of a session. In Subsection 5.A we present the scheme for the buffer organization and discuss the mechanism for the freezing and defreezing of capacity in the FIFO case, in Subsection 5.B we consider the buffer space required to ensure lossless transmission and efficient link utilization, and in Subsection 5.C we give the rate allocation algorithm at a switch.

A. Buffer organization

Each FIFO buffer is partitioned into $K+1$ bins, with capacities given by B_K, $B_0 = C\tau_0$ and $B_i = C\tau$, for $i = 1, 2, \ldots, K-1$ (see Fig. 4), where C is the maximum capacity of the link, and τ_0 and τ are both constants that are determined as described in Subsection 5.B (τ_0 and τ can be viewed as the time required for the corresponding bins to fill at the full rate).

Flow control starts when the buffer space occupied at $Q_{i+1}(n)$ exceeds $B_0 = C\tau_0$. When the buffer occupancy rises and crosses a bin a bin boundary B_j a THROTTLE packet is sent to the preceding node. The THROTTLE packet asks the preceding node s_i to reduce its output rate to $Q_{i+1}(n)$ by C/K. The capacity C/K is then temporarily *frozen*, by which we mean that it is not available for use by the sessions routed through $Q_{i+1}(n)$. As the buffer occupancy falls, each time that it crosses a bin boundary B_j a DEFREEZE packet is sent to the preceding node. The DEFREEZE packet informs the preceding node s_i that it can increase its output rate to $Q_{i+1}(n)$ by C/K. The capacity C/K is then *defrozen*, and is once again available for use by new or ongoing sessions routed through $Q_{i+1}(n)$. In other words, when a bin B_j at $Q_{i+1}(n)$ becomes full, a THROTTLE packet is sent to the previous node informing it that the frozen input capacity into $Q_{i+1}(n)$ should be increased to $\frac{i+1}{K}C$, and when a bin B_j at $Q_{i+1}(n)$ empties, a DEFREEZE packet is sent to the previous node informing it that the frozen input capacity into $Q_{i+1}(n)$ should be reduced to $\frac{i}{K}C$. Therefore, the buffer organization enables the throttling process to depend only on the level of buffer occupancy, and makes it independent of the particular way in which the input rates to the FIFO buffer change. In particular, node s_{i+1} sends at most K successive THROTTLE packets (that is, THROTTLE packets with no DEFREEZE packets sent in between) to node s_i when buffer occupancy rises and at most K successive DEFREEZE packets to node s_i when the buffer occupancy falls.

B. Buffer requirements

Consider a time t at which bin B_0 is full and the bins $B_1, B_2, \ldots, B_K$ are empty, and data arrives at $Q_{i+1}(n)$ at the (maximum) rate of C bits/sec. Assume also that no outgoing capacity is allocated to $Q_{i+1}(n)$. In this worst case scenario, buffer $Q_{i+1}(n)$ fills at rate C, and node s_{i+1} sends K THROTTLE packets to node s_i at τ second intervals. Buffer $Q_{i+1}(n)$ must therefore have space to store the packets that arrive at node s_{i+1} between time

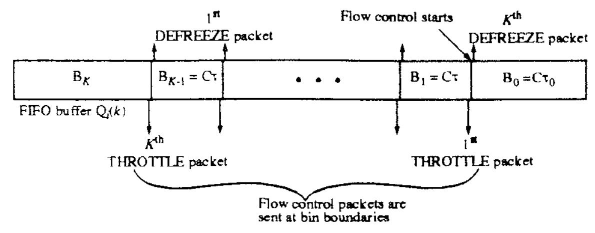
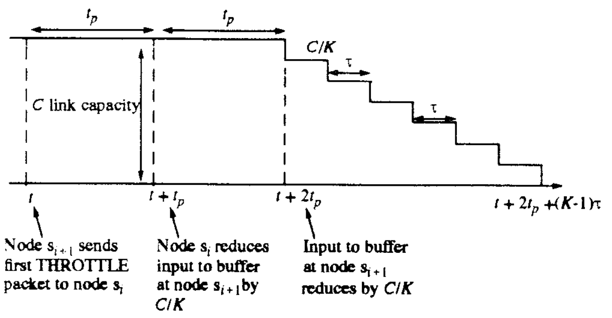


Fig. 4. Illustrates the way in which the buffer is partitioned into bins in the buffer partitioning scheme.



t_p
t_p
C/K
τ
τ
C link capacity
t
$t+t_p$
$t+2t_p$
$t+2t_p+(K-1)\tau$
Node s_{i+1} sends first THROTTLE packet to node s_i
Node s_i reduces input to buffer at node s_{i+1} by C/K
Input to buffer at node s_{i+1} reduces by C/K

Fig. 5. Illustrates the buffer requirements to ensure lossless transmission. The figure shows the variation with time of the input rate into $Q_{i+1}(n)$ as node s_i takes action in response to the THROTTLE packets sent by node s_{i+1}.

t and the time at which the throttling action taken at node s_i becomes effective at node s_{i+1}.

As indicated in Fig. 5, the total buffer space B needed at $Q_{i+1}(n)$ is

$$B = B_0 + 2t_pC + \frac{C(K-1)\tau}{2}. \qquad (4a)$$

Equivalently, dividing by link capacity C to express quantities in terms of the time parameters, we obtain

$$T = \tau_0 + 2t_p + \frac{(K-1)\tau}{2}. \qquad (4b)$$

The buffer space B_K is found as

$$\begin{aligned} B_K &= B - B_0 - \sum_{i=1}^{K-1} B_i \\ &= 2t_pC + \frac{C(K-1)\tau}{2} - C(K-1)\tau \\ &= 2t_pC - \frac{C(K-1)\tau}{2}. \qquad (5) \end{aligned}$$

The buffer space B_K is needed to store the data that arrives at $Q_{i+1}(n)$ between time $t+(K-1)\tau$ at which node s_{i+1} transmits the last THROTTLE packet, and time $t+2t_p+(K-1)\tau$, at which the inflow to $Q_{i+1}(n)$ ceases completely.

The parameters τ_0 and τ are chosen so that link capacity is used efficiently and does not remain idle unnecessarily. As shown in Fig. 6, this is guaranteed when τ_0

and τ are chosen so that

$$2t_p \leq \tau_0 + \frac{(K-1)\tau}{2}, \tag{6}$$

is satisfied.

One possible solution to Eq. (6) is to set

$$\tau_0 = t_p, \quad \text{and} \quad (K-1)\tau = 2t_p, \tag{7}$$

which gives a total buffer space equal to $B = 4t_pC$.

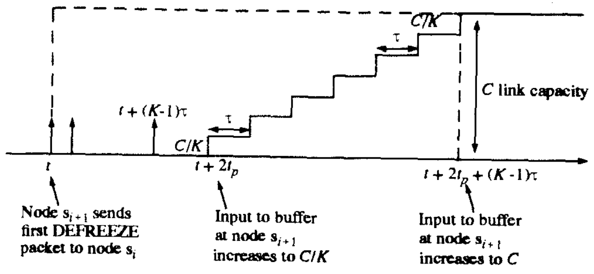


Fig. 6. Illustrates how the *defreeze* procedure works. The vertical axis corresponds to the input rate out of $Q_{i+1}(n)$. The worst case scenario, in terms of link utilization, corresponds to the case where at time t the bins $B_0, B_1, \ldots, B_{K-1}$ at buffer $Q_{i+1}(n)$ are full, the bin B_K is empty, and $Q_{i+1}(n)$ is granted capacity C. Buffer $Q_{i+1}(n)$ then starts emptying at the rate of C bits/sec, and sends at most K DEFREEZE packets to the preceding node s_i at τ second intervals. Therefore, as shown above, the input rate to buffer $Q_{i+1}(n)$ becomes equal to C/K at time $t+2t_p$ and increases to C at time $t+2t_p+(K-1)\tau$, and the outgoing link at node s_{i+1} remains continuously busy if the amount of data transmitted over the outgoing link at node s_{i+1} in the interval $[t, t+2t_p+(K-1)\tau]$ is less than the data that it had at time t and the data that it received in the interval $[t, t+2t_p+(K-1)\tau]$, that is if

$$C[2t_p + (K-1)\tau] \leq B_0 + \sum_{i=1}^{K-1} B_i + \frac{C(K-1)\tau}{2},$$

or equivalently $C[2t_p + (K-1)\tau] = B_0 + \frac{3C(K-1)\tau}{2}$.

In the Thunder and Lightning network switch, each FIFO buffer can hold up to $B = 750 \times 10^3$ packets. Since the link capacity is $C = 40$ Gbit/s or 94.34×10^6 packets/sec and the propagation delay is $d = 5\ \mu s$/km, the spacing L between two successive switches should satisfy

$$4dLC \leq B,$$

which for the above parameters gives $L = 398$ km. Since the need to regenerate data bits dictates that the inter-switch spacing be 100 km or less, the RGVC protocol does not impose any additional constraints on the design.

C. Rate allocation procedure

In the previous subsections we explained how the FIFO buffers in the RGVC protocol are partitioned and discussed the requirements on the buffer space to ensure lossless transmission. In this subsection we discuss how a node s_i allocates the transmission rates $R_i(k)$, to each of the FIFO data buffers feeding an outgoing link (see Fig. 7). (Recall, that in the Thunder and Lightning network, each link is fed by three FIFO data buffers.)

We define the supply $S_i(k)$ at FIFO k as

$$S_i(k) = \frac{|Q_i(n)|}{T_p}, \tag{8}$$

where $|Q_i(k)|$ is the buffer space occupied at $Q_i(k)$, and T_p is a parameter, which is at least as large as the time between successive executions of the rate allocation algorithm. The supply$S_i(k)$ can be viewed as the rate at which FIFO k should transmit to serve incoming traffic and clear the occupied buffer space within time T_p.

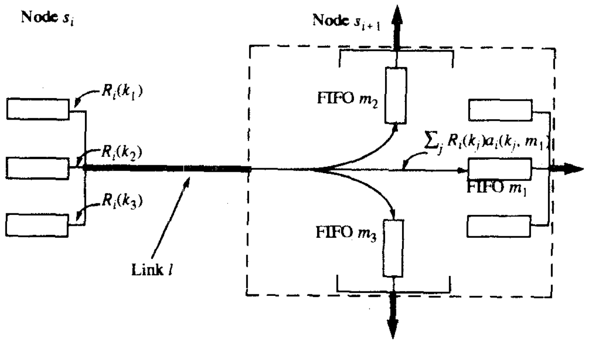


Fig. 7. Illustrates the data flows between the FIFO buffers at two successive nodes. The total outflow from a FIFO k at node s_i is equal to $\sum_m R_i(k)a_i(k,m)$, which is the sum of the rates at which data from FIFO k flows to the three FIFO buffers at node s_{i+1}. Similarly, the total inflow into a FIFO m at node s_{i+1} is equal to $\sum_k R_i(k)a_i(k,m)$, which is the sum of the rates at which data flows to FIFO m from the three FIFO's at node s_i.

The rate allocation algorithm at a node attempts to maximize the total outflow from the node, under the constraint that the sum of output rates $R_i(k)$, $k \in \{k_1, k_2, k_3\}$, is less than the capacity C of the outgoing link and that the input rate to each FIFO at the next node is less than what it can accept without buffer overflow. We let $a_i(k,m)$ be the fraction of the data output from a FIFO k at node s_i and destined for a FIFO m at node s_{i+1} (see Fig. 7). The fractions $a_i(k,m)$, $k \in \{k_1, k_2, k_3\}$, and m, $m \in \{m_1, m_2, m_3\}$ are obtained by periodically measuring the outflow from each FIFO buffer $Q_i(k)$, $k \in \{k_1, k_2, k_3\}$, at a node s_i. We also let $F_i(m)$ be the last estimate that node s_i has about the frozen input capacity of FIFO m at node s_{i+1}, and we formulate the rate allocation problem as the following linear programming problem:

(Problem P) $\quad \max \sum_{k=k_1}^{k_3} R_i(k)$

subject to

$$\sum_{k=k_1}^{k_3} R_i(k)a_i(k,m) \leq C - F_i(m), \text{ for all } m, \tag{9a}$$

$$\sum_{k=k_1}^{k_3} R_i(k) \leq C, \tag{9b}$$

and

$$R_i(k) \leq S_i(k), \text{ for } k\eta\{k_1, k_2, k_3\}, \tag{9c}$$

where $S_i(k)$ is given by Eq. (8).

Problem P is solved whenever one of several events occurs. It is solved either when $F_i(m)$ changes for some m (which does not happen more often than τ seconds), or when the fractions $a_i(k, m)$ change significantly. If none of these events happens, problem P is still solved at intervals of T_p seconds.

If one of the FIFO's, say FIFO k_1, is a source of a session $\mathcal{S}$ it is given a loweest priority when allocating the rates. This is done by assuming $R_i(k_1) = 0$ and solving problem P only for the two variables $R_i(k_2)$ and $R_i(k_3)$ that remain, and then finding

$$n = \text{argmin}_{m \in N_S}(C - F_i(m)),$$

where $N_{\mathcal{S}}$ is the set of FIFO's at node s_{i+1} through which session $\mathcal{S}$ is routed, and setting

$$R_i(k_1) = \min(C - F_i(n), C - \sum_{j=k_2}^{k_3} R_i(j)), \tag{10}$$

The reason for doing this is that reducing $R_i(k_1)$ for a session orginating at a node can be done easily and has less severe effects on the network than reducing $R_i(2)$ and $R_i(3)$ (it does not require the transmission of THROTTLE packets and the freezing of capacity at other other nodes).

VI. Concluding remarks

We presented the main features of the RGVC connection control protocol, assuming both RAM and FIFO buffers at the switches. We introduced the concept of *freezing* of capacity and demonstrated how it leads to the lossless character of the RGVC protocol. We argued that the RGVC protocol leads to substantially lower connection setup times than WRVC protocols. It also improves capacity utilization (if the required capacity is available), since it reserves capacity for less time than WRVC protocols. Also, instead of rejecting a session if the required rate is unavailable, it allows the session to establish a connection at a lower rate, which can be increased to the desired rate when capacity is freed. The RGVC protocol is being implemented in the Thunder and Lightning network being built at UCSB. This implementation will enable us to test the operation of the protocol, study its performance, and validate and refine its features, to make the protocol a viable candidate for network protocols in future high-speed networks.

References

[1] S. Butner and D. Skirmont, "Architecture and design of a 40 Gigabit per second ATM switch", in *Proc. Int'l Conference on Computer Design*, Austin, TX, Oct. 1995.

[2] E. A. Varvarigos and V. Sharma, "An efficient reservation connection control protocol for gigabit networks", submitted *IEEE/ACM Trans. on Networking*, January 1995.

[3] I. Cidon and I. S. Gopal, "PARIS : An approach to integrated high-speed private networks", *Int'l J. Digital and Analog Cabled Systems*, vol. 1, no. 2, pp. 77–86, Apr-June 1988.

[4] D. L. Tennenhouse and I. M. Leslie, "A testbed for wide area ATM research", in *Proc. ACM SIGCOMM'89*, Austin, TX, September 1990, pp. 182–190.

[5] I. Cidon, I. Gopal, P.M. Gopal, R. Guérin, et al., "The plaNET/ORBIT high-speed network", *Journal of High Speed Networks*, vol. 2, no. 3, pp. 171–208, 1993.

[6] D. D. Clark, B. S. Davie, D. J. Farber, and I. S. Gopal, "The AURORA gigabit testbed", *Computer Networks and ISDN Systems*, vol. 58, pp. 599–621, May 1970.

[7] M. Devault, J. Y. Cochennec, and M. Servel, "The "Prelude" ATD experiments : assessments and future prospects", *IEEE J. Selected Areas Commun.*, vol. 6, no. 9, pp. 1528–1537, December 1988.

[8] D. J. Greaves, D. Lioupis, and A. Hopper, "The Cambridge backbone ring", in *Proc. IEEE Infocom90*, San Fransisco, CA, June 1990, pp. 8–14.

[9] B. Butscher and J. Kanzow, "The BERKOM project: a B-ISDN field trial in Berlin", in *Proc. 10th Int'l Conference on Computer Communication*, New Delhi, India, Nov. 1990, pp. 4–9.

[10] B. Pehrson and S. Pink, "MultiG: a Swedish research program on multimedia applications in gigabit networks", in *First Int'l Workshop on Network and Operating System Support for Audio and Video*, Berkeley, CA, November 1990.

[11] B. Pehrson and L. Gauffin, "MultiG distributed multimedia applications and gigabit networks", in *Proc. Int'l Switching Symp.*, Yokohama, Japan, 25-30 October 1990.

[12] A. Segall and J. M. Jaffe, "Route setup with local identifiers", *IEEE Trans. on Comm.*, vol. 34, no. 1, pp. 1019–1028, January 1986.

[13] J. Y. Hui, "Resource allocation for broadband networks", *IEEE J. Selected Areas Commun.*, vol. 6, no. 9, December 1988.

[14] I. Cidon, I. S. Gopal, and R. Guérin, "Bandwidth management and congestion control in plaNET", *IEEE Communications Magazine*, vol. 29, no. 10, pp. 54–64, October 1991.

[15] P. E. Boyer and D. P. Tranchier, "A reservation principle with applications to the ATM traffic control", *Computer Networks and ISDN Systems*, vol. 24, no. 4, pp. 321–324, May 1992.

[16] D. P. Tranchier, P. E. Boyer, Y. M. Rouaud, and J. Y. Mazeas, "Fast bandwidth allocation in ATM networks", in *Proc. Int'l Switching Symposium*, Tokyo, Japan, 25-30 October 1992.

[17] I. Cidon, I. S. Gopal, and A. Segall, "Connection establishment in high-speed networks", *IEEE/ACM Trans. on Networking*, vol. 1, no. 4, pp. 469–481, August 1993.

[18] L. Gun and R. Guérin, "Bandwidth management and congestion control framework of the broadband network architecture", *Computer Networks and ISDN Systems*, vol. 26, no. 1, pp. 61–78, September 1993.

[19] C. Partridge, "Protocols for high-speed networks: some questions and a few answers", *Computer Networks and ISDN Systems*, vol. 25, no. 9, pp. 1019–1028, June 1993.

[20] E. A. Varvarigos and V. Sharma, "Ready-to-go virtual circuit – a loss-free connection control protocol for gigabit networks", in preparation.