RZ 2754 (# 89232) 10/30/95
Computer Science/Mathematics 13 pages

Research Report

A General and High-Speed Rate Control Scheme

Willibald Doeringer* and Marco Heddes

IBM Research Division
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

*Present address: FH Worms, Erenburger Str. 19, D-67549 Worms, Germany

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

A General and High-Speed Rate-Control Scheme

Willibald Doeringer* and Marco Heddes

IBM Research Division, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

This article describes a basic, simple mechanism for the rate control of data transmission and reception which can be efficiently implemented in hardware to operate at gigabit-per-second media speeds, and from which all relevant schemes described in the literature can be readily constructed.

*Present address: FH Worms, Erenburger Str. 19, D-67549 Worms, Germany

1 Introduction

The tremendous increase in recent years of the available bandwidth in broadband networks has drawn new attention to the problem of congestion control [1]. A problem which has become more complicated due to the dominance of propagation delay over transmission time which severely impacts the usefulness of end-to-end feedback controls, the vast amount of user data which may be transiting broadband networks at any given point in time, the expected burstiness of the traffic generated by end users [2], and the potentially immense number of individual data streams which have to be managed, controlled and supplied with network resources [3, 4]. Hence, it has been widely argued that even for basic congestion control in broadband networks, rate control of individual data streams has to be exercised at the network access points, commonly referred to as *access control* or *policing* [5]. The objective here is to limit a sender in its transmission of packets, such that the network does not become overloaded.

Also, the traditional window-based flow-control schemes have been shown to result in very low utilization of high-speed links, mainly due to premature credit depletion since the maximum useful window sizes are too small to permit transmission of a round-trip time's equivalent of data [6, 7]. As a consequence, designers of recent (lightweight) protocols have also accepted the need to incorporate rate-based flow-control schemes to optimize the end-to-end throughput in a high-speed environment [7].

However, all rate control schemes proposed in the literature[1], as well as the very few published proposals for hardware implementations thereof have several drawbacks:

Lack of generality. Current proposals for rate control in broadband networks do not isolate a common basic and simple mechanism from which other relevant schemes can be easily constructed.

Complexity and cost. The lack of a well-defined basic building block made hardware designers focus on the implementation of specific rate-control algorithms to avoid complex structures with their attendant cost and performance problems. Hence, current communication systems support only very limited combinations of rate control schemes.

Periodic updates. All documented hardware implementations require periodic operations for each rate control instance, necessitating dedicated hardware resources such as counters and processing elements [8]. This approach clearly limits the number of such instances that can be placed on a single chip and hence the number of data streams that may be controlled by such a device.

This article attempts to improve on the current situation: The next section presents the design of a versatile, basic rate control mechanism, which is demonstrated to apply well to all relevant rate-control schemes discussed in the literature[2]. Then follows the description of an abstracted hardware implementation of the proposed rate-control core, and the subsequent discussion of the performance of the most complex scheme shows that operation at Gigabit-per-second media speeds may be readily obtained with current VLSI technology.

[1] They are presented on p. 4.

[2] A preliminary version of this article has appeared in [9, 10].

2 The Core Mechanism

Fig. 1 depicts a general rate-controlled communication system: On its way from a sender to the transmission medium, (or from the transmission media to the receiver), a packet, or cell[3], passes one (or possibly multiple) rate-control instance(s). Each such instance implements a particular rate-control scheme. As different as all relevant schemes presented in the literature may seem at first sight, they can all be viewed as *credit-based control systems*. The transmission or reception of one packet consumes one *credit*, and is only permitted if the credits are not fully depleted. Any rate control scheme may hence be described by the initial credit, the update period for credit replenishment, and the algorithm for calculating the new credits — with the convention that the concept of credits, or *tokens*, is used in a purely descriptive sense and implies no particular flow-control scheme.

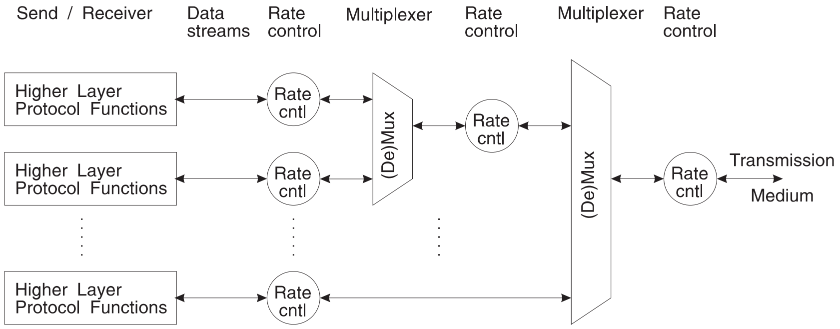


Figure 1: Conceptual view of packet transmission and rate control.

In order to avoid costly periodic operations for the implementation of a rate-control scheme, we propose a functionally equivalent but strictly demand-driven approach. Appropriate (cumulative) credit updates are only performed when a rate-control-instance processes a packet, that is, at the *packet events*. Whenever enough credit is available, packets are processed and credits are updated accordingly, thus establishing the so-called *successful packet events*. Otherwise, a credit-depletion signal from a rate-control-instance may cause a communication system to perform one of the following actions: (i) The packet is discarded without further notice. (ii) The packet is marked, such that during its travel through the network it has a higher chance of being discarded when network nodes experience congestion. Or (iii), packet transmission is delayed until an appropriate *wait-time* has elapsed such that enough credit is available when the packet is re-scheduled.
In summary, the key to obtain a high-speed implementation is that rate-control instances need to become active only at their packet events. Consequently, their aggregate processing capacity is dictated only by the media speed, and no longer depends upon the number of active data streams.

2.1 The Credit Function

In our proposed scheme, the new credit becomes a function of the previous credit(s) and the time(s) at which the previous packet events occurred. This approach is formalized in

[3]We use the term *packet* in a generic sense, including variable-length and fixed-size versions.

the subsequent definition of the basic rate-control mechanism, using the following notational conventions:

t The system time, measured in appropriate units. For simplicity of presentation, we assume that rate-control instances are always initialized at time $t = 0$. In practice, one needs to relate a central clock to the relative time scales of the instances by a straightforward adjustment based on the respective times of instance creation.

i The number of packet events. Initially, $i = 0$. At each packet event, i is incremented, thus providing a numbering scheme for the packets.

T_i The time of the i-th packet event; T_0 is defined as zero.

C_i The credit available at the i-th packet event. The packet event is successful, iff $C_i \geq 1$.

Δ_i The wait-time at packet event i. $\Delta_i \leq 0$, iff the i-th packet event has been successful. Otherwise, $T_i + \Delta_i$ denotes the earliest time at which the packet may be transmitted or received. The wait-time is only relevant for communication systems where packet transmission may be delayed. It is ignored when packets are discarded or simply marked during credit depletion.

$\vec{a}_i$ A vector of auxiliary state variables and (configuration) parameters at time T_i. $\vec{a}$ depends on the particular rate-control scheme, and typically consists of a few configuration parameters as exemplified in the next section.

CF The *credit function* defines how the new credit is calculated. It derives C_{i+1}, Δ_{i+1} and $\vec{a}_{i+1}$ from the previous C_i's, T_i's, and $\vec{a}$'s with $0 \leq i \leq n$. More formally, for a packet event $n+1$ at time $t > T_n$ we have in the most general case: $(C_{n+1}, \Delta_{n+1}, \vec{a}_{n+1}) = CF(t, (C_n, T_n, \vec{a}_n), \ldots, (C_0, T_0, \vec{a}_0))$. Of the various credit functions described in the next section, most are 'markovian' in that only C_n, T_n and $\vec{a}_n$ are relevant.

The conceptional operation of the resulting rate controller is modeled by the following pseudo code for a rate-control instance, which when presented with a packet at time t, returns zero when the packet may be processed, or the wait-time otherwise.

Rate Control Mechanism Pseudo-Code

$T_{n+1} = t$ /* Note the current time. */
Calculate C_{n+1}, Δ_{n+1} and $\vec{a}_{n+1}$ according to the credit function CF
/* based on the history until time T_n. */
if ($C_{n+1} \geq 1$) /* Successful event! Accept the packet.*/
  $C_{n+1} = C_{n+1} - 1$ /* Processing a packet consumes one credit. */
$\Delta = \max(0, \Delta_{n+1})$ /* Set the return value. */
$n = n + 1$ /* Advance the counter of packet (events). */
return(Δ) /* Processing may proceed if zero. */

In cases where a non-zero wait-time[4] is returned, a timer may be started which expires when enough credit is available and the packet may be re-scheduled. Ref. [10] describes an efficient method for timer management with a latency of all timer operations of O(1), independent of the number of active timers.

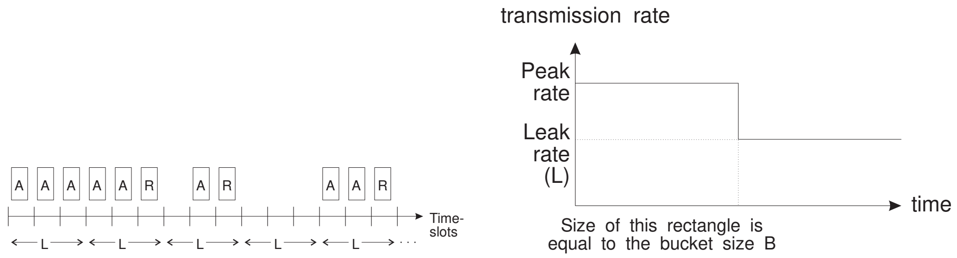
[4]Formally $\Delta_i = \min\{\epsilon \geq 0 \mid C \geq 1 \ \mathit{with}\ (C, T, \vec{a}) = CF(T_i + \epsilon, (C_{i-1}, T_{i-1}, \vec{a}_{i-1}), \ldots)\}$.

3 Applying the Core Mechanism

This section discusses the relevant rate-control schemes proposed in the literature and defines for each a suitable credit function to integrate the respective algorithm into the general concepts described in the previous section. For ease of presentation a slotted transmission system is assumed, that is, the packets are assumed to be of equal length and to be transmitted or received in one time slot. Extensions to variable-length packets or an absolute time frame, such as milliseconds, are either straightforward or will otherwise be explained as appropriate. The presented concept of a versatile, basic rate control scheme permits the construction of considerably more schemes than are discussed below. However, the analysis of the benefits of such yet to be defined schemes over current ones is a topic in its own and will not be pursued in this article.
In the following figures, $\boxed{\text{A}}$ indicates that a packet has been accepted or successfully transmitted, and $\boxed{\text{R}}$ denotes a packet whose processing has been rejected in the respective time slot due to credit depletion. For the description of the various credit functions we focused on the definition of CF, i.e., the function proper, and omitted the standard trailing sequence of the above full processing which checks for $C_{n+1} \geq 1$ and decrements its current value for successful packet events.

3.1 Leaky Bucket

The *leaky bucket* mechanism [8] is a credit-based system where the transmission or reception of each packet consumes one credit, and where credits are replenished at a given rate $1/\mathcal{L}$, the *leak* or *drain rate*, up to a maximum total number of $\mathcal{B}$, the *bucket size*. Transmission or reception of packets is only permitted if the credits are not depleted. As a result, a leaky-bucket controlled system may send or receive a maximum burst of $\mathcal{B}$ back-to-back packets and use 1 out of $\mathcal{L}$ slots in the long-term average (see Fig. 2(a)). The effects of leaky-bucket policing can



(a) Leaky bucket with $\mathcal{L}=3$ and $\mathcal{B}=4$.

(b) Effects of leaky-bucket policing

Figure 2: Leaky bucket mechanism and effects.

be nicely demonstrated by its impact on a sender with an infinite queue of packets which it attempts to transmit at maximum speed. An approximation of the resulting transmission rate is depicted in Fig. 2(b). In the general context of credit functions, the leaky-bucket control scheme can be described with a credit function given by:

$$\begin{cases} C_{n+1} &= \min(\mathcal{B},\ C_n + (t - T_n)\ /\ \mathcal{L}) \quad \text{with} \quad C_0 \ = \ \mathcal{B} \\ \Delta_{n+1} &= T_n + \mathcal{L} - t \\ \vec{a}_{n+1} &= (\mathcal{B}, \mathcal{L}) \end{cases}$$

The basic leaky bucket scheme does not control the peak rate. Hence bursts of packets may occur which transmit the full size of a bucket at medium speed. Such burstiness is, however, quite unfavorable for any optimization of network resource allocation [11, 12]. Hence it appears attractive to control the peak rate also, especially since the broadband networks (plan to) offer variable bit-rate access at the user-network interfaces [13, 14]. A pertinent rate-control scheme may be constructed out of a pair of cooperating, e.g. sequential, leaky-bucket control instances. One of them controls the average rate as described above, and the other operates with a different set of parameter values to control the peak rate. Typically the peak rate controller will run with a small bucket size and high leak rate to achieve a sufficiently fine granularity of control, whereas the average rate controller will employ a larger bucket size and a significantly smaller leak rate. The mechanism is illustrated in Fig. 3 with a bucket size of $\mathcal{B}_A = 4$, a leak rate defined by $\mathcal{L}_A = 3$ for the control of the average rate, and of $\mathcal{B}_P = 1$ and $\mathcal{L}_P = 2$ for the peak rate control.

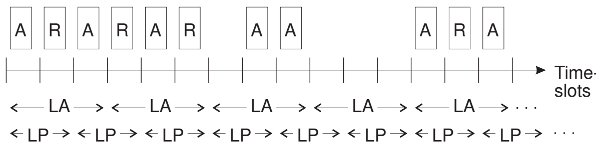


Figure 3: Peak-rate control.

3.1.1 The ATM Forum Generic Cell-Rate Algorithm

Recently, the ATM Forum voted that in ATM networks the Available Bit-Rate Service [15] should be flow-controlled by means of the Generic Cell-Rate Algorithm [16] — a so-called 'continuous-state leaky bucket scheme'.
For every connection, the proposed algorithm maintains state variables LCT and X, where LCT denotes the time of the last successful packet event, the 'Last Conformance Time', and X the current balance in the account of utilized time units. Each cell reception (or transmission) is debited onto that account as I units, while the idle time since LCT is credited, that is, subtracted from the debt — all within a lower bound of zero and an upper bound for the maximum debt of L. At each cell arrival, the decision whether to accept the cell, and the updates of the state variables are performed according to the following operations, starting with a value of zero for both X and LCT at the establishment of a connection.

```
X' = X − (t − LCT)  /* Credit the idle time. */
if (X' < 0) X' = 0  /* Lower bound is zero. */
if (X' ≤ L)         /* Accept the cell, and update X and LCT. */
    X = X' + I          /* Charge I units per cell. */
    LCT = t             /* Adjust time of last successful packet event */
```

As is shown in the appendix, the flow-control policy determined by a Generic Cell-Rate Algorithm is equivalent to that of the above standard leaky bucket scheme with configuration parameters $\mathcal{B} = 1 + L/I$ and $\mathcal{L} = I$.

3.2 Window-Based Schemes

Window-based schemes all define a fixed-length time interval of $\mathcal{W}$ time slots, the *window*, and the maximum number of packets, $\mathcal{M}$, which may be transmitted or received during such windows, the *credit limit*. Variations result from the definition when a new window starts as discussed below. Windows are numbered beginning with 1.

3.2.1 Jumping Window

With this scheme, windows are non-overlapping, contiguous time intervals, resulting in a policing behavior as shown in Fig. 4.

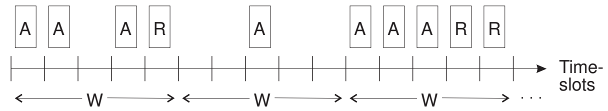


Figure 4: Jumping window with $\mathcal{W}=5$ and $\mathcal{M}=3$.

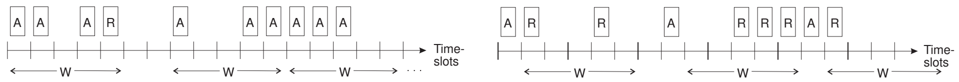
The credit function is given by:

$$\begin{cases} C_{n+1} &= \text{if } ((t \text{ div } \mathcal{W}) \neq (T_n \text{ div } \mathcal{W})) \text{ then } \mathcal{M} \text{ else } C_n \quad \text{with} \quad C_0 = \mathcal{M} \\ \Delta_{n+1} &= t \bmod \mathcal{W} \\ \vec{a}_{n+1} &= (\mathcal{W}, \mathcal{M}) \end{cases}$$

For $t > T_n$, the condition $(t \text{ div } \mathcal{W}) \neq (T_n \text{ div } \mathcal{W})$ implies that a new window has started and hence the maximum number of credits is available, whereas only C_n credits are at disposal otherwise. If an implementation uses a system clock with period $\mathcal{W}$, the div operation can be omitted. An obvious extension to variable-length packets and an absolute time frame yields the *burst control* scheme defined for recent transport protocols [17].

3.2.2 Triggered Window, Interpacket Gap

Here, windows are non-overlapping time intervals of equal length which are, however, not necessarily contiguous. A new window starts after the previous one, but only during the processing of the next packet event [18]. The operation of this scheme is exemplified in Fig. 5(a).



(a) Triggered Window with $\mathcal{W}=5$ and $\mathcal{M}=3$.

(b) Interpacket gaps with $\mathcal{W}=5$.

Figure 5: Triggered Window and Interpacket Gaps.

To fit this scheme into our model, a state variable τ_n is required, which represents the beginning of the last window according to the following definition.

$$\begin{cases} \tau_{n+1} = \text{if } (T_{n+1} - \tau_n \geq \mathcal{W}) \text{ then } T_{n+1} \text{ else } \tau_n \quad \text{with} \quad \tau_0 = T_0 \end{cases}$$

The credit function is then given by:

$$
\begin{cases}
C_{n+1} & = & \text{if } (t - \tau_n \geq \mathcal{W}) \text{ then } \mathcal{M} \text{ else } C_n \quad \text{with} \quad C_0 \;=\; \mathcal{M} \\
\Delta_{n+1} & = & T_n + \mathcal{W} - t \\
\vec{a}_{n+1} & = & (\tau_{n+1}, \mathcal{W}, \mathcal{M})
\end{cases}
$$

A particular variant are the so-called *interpacket gaps* [19], which operate with $\mathcal{M} = 1$ and start a new window after the previous one and *after* the transmission of the next packet (Fig. 5(b)). As becomes particularly apparent with variable-length packets, this scheme is not primarily intended to smooth traffic, but to enforce sufficient time gaps between successive transmissions to prevent receivers from being overrun by back-to-back packets.

3.2.3 Moving Window

This scheme aims at the emulation of a reserved circuit with bandwidth $\mathcal{M}/\mathcal{W}$ [19, 20]. Each successful packet event triggers the start of a new window, possibly overlapping with still active ones, thus guaranteeing a further smoothing effect (Fig. 6). The windows are 'gliding' along the time axis and state has to be kept about the last $\mathcal{M}$ such windows. To model this

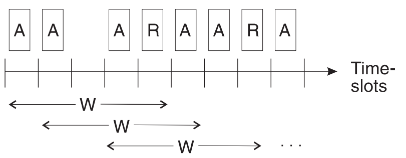


Figure 6: Moving window with $\mathcal{W}=5$ and $\mathcal{M}=3$.

scheme as a credit-based system, state variables $\tau_{n,j}$ are required[5], which denote the starting times of the last $\mathcal{M}$ windows, i.e., of the last $\mathcal{M}$ successful packet events. These variables obey the following recursive definition.

$$
\begin{cases}
\tau_{n+1,n+1} & = & \text{if}(\tau_{n,n-(\mathcal{M}-1)} \leq T_{n+1} - \mathcal{W}) \text{ then } T_{n+1} \text{ else } \tau_{n,n}, \quad \tau_{0,0} = T_0 \\
\tau_{n+1,j} & = & \text{if}(\tau_{n,n-(\mathcal{M}-1)} \leq T_{n+1} - \mathcal{W}) \text{ then } \tau_{n,j} \text{ else } \tau_{n,j-1}, \quad n+1-(\mathcal{M}-1) \leq j \leq n \\
\tau_{n+1,j} & = & -\infty, \text{ for } j < 0,
\end{cases}
$$

The credit function can then be defined as:

$$
\begin{cases}
C_{n+1} & = & \left(\sum_{i=n-\mathcal{M}+1}^{n} (\text{if } (\tau_{n,i} \leq t - W) \text{ then } 1 \text{ else } 0)\right) \quad \text{with} \quad C_0 = \mathcal{M} \\
\Delta_{n+1} & = & T_{n-(\mathcal{M}-1)} + \mathcal{W} - t \\
\vec{a}_{n+1} & = & (\tau_{n+1,n+1}, \tau_{n+1,n}, \ldots, \tau_{n+1,n+1-(\mathcal{M}-1)}, \mathcal{M}, \mathcal{W})
\end{cases}
$$

It is interesting to note, that $C_{n+1} \geq 1$ iff $\tau_{n,n-(\mathcal{M}-1)} \leq t - \mathcal{W}$. Hence, the decision whether to process a packet can be based solely on the oldest of the last $\mathcal{M}$ timestamps.

3.2.4 Exponentially-Weighted Moving Average

In contrast to the previous window schemes, here the maximum number of packets which may be transmitted or received during one window is a function of the number of packet events

[5]for $n-(\mathcal{M}-1) \leq j \leq n$

in the previous window [19, 20]. The algorithm specifies the (variable) credit limits for the individual windows as follows. In the p-th window, let S_p denote the so-called *weighted sum*, and X_p the *total number of packet events*. Then the following recursive relation is defined for a given *weight factor* $\alpha \in [0,1]$:

$$S_p = (1-\alpha)X_p + \alpha S_{p-1}, \quad 1 \leq p, \quad S_0 = \mathcal{M} \tag{1}$$

Packet events in window p are successful, i.e., their transmission is completed or their arrival is accepted, only up to a maximum number of S_{p-1} such events - i.e., S_{p-1} denotes the credit limit for the p-th window. One may observe, that the choice of $\alpha = 1$ yields the jumping window mechanism, and that $\alpha = 0$ allocates a credit limit for the current window equal to the number of packet events in the previous one, i.e., $S_{p-1} = X_{p-1}$. An example of this scheme is shown in Fig. 7 with α set equal to 1/2 and $\mathcal{M} = 3$.

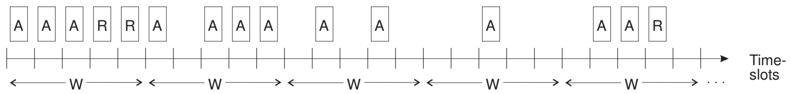


Figure 7: Moving average with $\mathcal{W}=5$, $\mathcal{M}=3$ and $\alpha=1/2$.

To derive a credit function, let $w(t)$ denote the number of the window[6] at time t, i.e., $w(t) = (t \ div \ \mathcal{W})+1$. Using the auxiliary variables B_n defined by

$$\left\{ \begin{array}{lcl} B_{n+1} & = & 1-\alpha + B_n \alpha^{w(T_{n+1})-w(T_n)} \quad \text{with} \quad B_0 \;=\; \alpha\mathcal{M} \end{array} \right.$$

we arrive at the following real-valued, non-negative credit function. The formal derivation of this result has been deferred to the appendix on page 10.

$$\left\{ \begin{array}{lcl} C_{n+1} & = & \text{if } (w(T_n) < w(T_{n+1})) \text{ then } \alpha^{w(T_{n+1})-1-w(T_n)} B_n \text{ else } C_n \quad \text{with} \quad C_0 = \mathcal{M} \\ \Delta_{n+1} & = & \mathcal{W} \;-\; (t \bmod \mathcal{W}) \\ \vec{a}_{n+1} & = & (B_{n+1}, \mathcal{M}, \mathcal{W}) \end{array} \right.$$

4 Implementation

The purpose of this section is to demonstrate that all operations for the credit functions are simple and easy to implement. Where the time slots are in the microsecond range, corresponding, for example, to ATM transmission rates of about 100 MBit/sec, a software implementation becomes feasible on a state-of-the-art microprocessor. For Gigabit/sec speeds, special hardware support is required, in particular for the most complex case of a controller which is fully programmable and supports the whole set of rate control schemes discussed here.
A straightforward high-speed hardware implementation of a general rate-controller may be composed of memory and a processing device as follows:

Memory: Upon creation of a rate control instance, a block of memory is allocated by any conventional memory manager to store a small set of variables local to the given instance, and all variables are initialized appropriately. To the controller a rate control instance is identified by the pointer to its memory block. Upon termination of the rate control instance, the memory is released.

[6] Window numbering starts with 1.

Controller: The controller incorporates a processing unit which implements one or potentially several different credit functions. In the latter case, the choice of the respective function to be employed for a given rate control instance is part of the instance's local variables stored in its memory block. Thus the rate control scheme may be changed on the fly between different packets in accordance to the scheme selected for the respective data stream.

The controller can be built as a basic programmable micro-processor with current chip technology. Its architecture will include a few registers, an arithmetic unit which may be optimized for the algorithms implemented by the credit functions, and a load-store memory interface. A conceptual example of the operation of such a device is discussed and evaluated in the following section.

4.1 Performance and Cost

In order to obtain an estimate of the required performance of a programmable high-speed rate-controller, we analyze the processing load for the most complex scheme, the exponentially weighted moving average. The analysis is based on the following example code sequence.

```
local variables
    dWindowSize      /* Window size, 𝒲 */
    rAlpha           /* Weight factor α ∈ [0, 1] */
    rBucketSize      /* B_n. Initialized to α𝓜 */
    dTimeStamp       /* T_n. Initialized to 0. */
    rCredit          /* C_n. Initialized to 𝓜 */

code
    dTimeStamp = t /* T_{n+1} */
    dWDif = (t / dWindowSize) - (dTimeStamp / dWindowSize)
    if (dWDif > 0)
        rCredit = rBucketSize * rAlpha↑(dWDif-1)
        rBucketSize = rAlpha * rCredit
    fi
    rBucketSize = rBucketSize + 1 - rAlpha
    if (rCredit ≥ 1)
        rCredit = rCredit - 1
        return(0)
    else
        return (dWindowSize - t mod dWindowSize)
    fi
```

To exploit the inherent parallelism in a hardware implementation, the above processing may be split into instructions for memory access and those for arithmetic calculations. Observing that the latter dominate, and assuming that each numerical operation completes in one clock-cycle, the full code sequence completes in 20 cycles[7]. Hence, an example clock cycle of 20 nsec, corresponding to a frequency of 50 MHz, yields a processing time of 400 nsec per packet. For ATM with a 53 bytes cell size, this corresponds to a data rate of 1 Gigabit/sec. Higher speeds can easily be obtained by shorter clock cycles or by use of special instructions, such as for the calculation of dWDif.

[7] All the other schemes execute in fewer cycles. E.g., 10 for the leaky bucket.

A Proofs

This section presents the proofs of the assertions made in the preceding text about the Generic Cell-Rate Algorithm proposed by the ATM Forum, and the Exponentially Moving Average scheme.

A.1 The ATM Forum Generic Cell-Rate Algorithm

It is claimed above that a Generic Cell-Rate Algorithm (GCRA) with configuration parameters L and I is equivalent to a standard leaky-bucket scheme with $\mathcal{B} = 1 + L/I$ and $\mathcal{L} = I$. That is, for any sequence of cell arrivals, both schemes pass the same decisions on whether to accept or reject an arriving cell. For a proof of this assertion, we first relate the sequence of X-values defined by the GCRA to those generated by a modified but equivalent algorithm, the latter of which is then shown to be equivalent to the respective leaky-bucket scheme in the above sense.

Let $[a]^+$ denote $\max(0, a)$ for any real number a, and let an arbitrary but fixed sequence of cell arrival times be given by $(T_n)_{n\geq 1}$, with $0 \leq T_1 < T_2 < \cdots < T_n < T_{n+1} < \cdots$. Then the GCRA flow-control policy can be implemented based on the sequence $(X_n)_{n\geq 0}$:

Generic Cell-Rate Algorithm (GCRA)

$X'_{n+1} = [X_n - (T_{n+1} - LCT_n)]^+$, with $X_0 = 0$ and $LCT_0 = 0$,
if ($X'_{n+1} \leq L$) /* Accept the cell and update. */
  $X_{n+1} = X'_{n+1} + I, \;\; LCT_{n+1} = T_{n+1}$
else /* Reject the cell. No changes. */
  $X_{n+1} = X_n, \;\; LCT_{n+1} = LCT_n$

where LCT_i denotes the time of the last successful packet event until time T_i. Since we intend to relate the GCRA, which adjusts its state variables only at successful events, to the standard leaky-bucket scheme with its continuous updates, we define an 'intermediary' flow-control policy which performs incremental updates at each cell arrival time based on values $(Y_n)_{n\geq 0}$ generated by the following algorithm.

Continuous Generic Cell-Rate Algorithm (conGCRA)

$Y'_{n+1} = [Y_n - (T_{n+1} - T_n)]^+$, with $Y_0 = 0$ and $T_0 = 0$,
if ($Y'_{n+1} \leq L$) /* Accept the cell and update. */
  $Y_{n+1} = Y'_{n+1} + I$
else /* Reject the cell and update. */
  $Y_{n+1} = Y'_{n+1}$

We may immediately infer the equivalence of the two flow-control schemes once we establish for $n \geq 0$ that $Y'_{n+1} = X'_{n+1}$ holds, since the decisions to accept or reject a cell are based solely on the values of Y'_{n+1} rsp. X'_{n+1}. This equality may be derived from the more general result (2) below by observing that for $b \geq 0$ and arbitrary a we have $[a^+ - b]^+ = [a - b]^+$, so that we may conclude

$$\begin{aligned} X'_{n+1} &= [X_n - (T_{n+1} - LCT_n)]^+ = [X_n - (T_n - LCT_n) - (T_{n+1} - T_n)]^+ \\ &= [[X_n - (T_n - LCT_n)]^+ - (T_{n+1} - T_n)]^+ \overset{(2)}{=} [Y_n - (T_{n+1} - T_n)]^+ = Y'_{n+1} \; \blacksquare \end{aligned}$$

The above derivation made use of the identity

$$Y_n = [X_n - (T_n - LCT_n)]^+, \quad \text{for } n \geq 0 \tag{2}$$

which holds for $n = 0$ by definition. Assuming it does so for all $0 \le i \le n$, we get in the case of a successful event for GCRA at time T_{n+1}, that $LCT_{n+1} = T_{n+1}$ and by induction

$$L \ge X'_{n+1} = [\ [X_n - (T_n - LCT_n)]^+ - (T_{n+1} - T_n)]^+ = [Y_n - (T_{n+1} - T_n)]^+ = Y'_{n+1}.$$

However, $Y'_{n+1} \le L$ implies $Y_{n+1} = Y'_{n+1} + I$, so that we arrive at

$$0 \ \le \ Y_{n+1} \ = \ Y'_{n+1} + I = X'_{n+1} + I \ = \ X_{n+1} = [X_{n+1} - (T_{n+1} - LCT_{n+1})]^+.$$

Whereas for an unsuccessful event, where by definition $LCT_{n+1} = LCT_n$ and $X_{n+1} = X_n$, we get $L < X'_{n+1} = Y'_{n+1}$ again by induction. Since $L < Y'_{n+1}$ implies $Y_{n+1} = Y'_{n+1}$, we may finally infer that Eq. (2) holds also in this case since

$$Y_{n+1} \ = \ Y'_{n+1} = X'_{n+1} \ = \ [X_n - (T_{n+1} - LCT_n)]^+ \ = \ [X_{n+1} - (T_{n+1} - LCT_{n+1})]^+ \ \blacksquare$$

The last step of our proof establishes the equivalence of the conGCRA discipline with that of a standard leaky bucket scheme with parameters $\mathcal{B} = 1 + L/I$ and $\mathcal{L} = I$. It suffices to show that the respective sequences $(Y_n)_{n\ge 0}$ and $(C_n)_{n\ge 0}$ obey the relation

$$I \times \min(\mathcal{B}, C_n + (T_{n+1} - T_n)/I) \ = \ L + I - Y'_{n+1} \quad \text{for } n \ge 0 \tag{3}$$

since this equality implies the following equivalences for $n \ge 0$,

The leaky bucket scheme accepts a cell at time T_{n+1}	$\Leftrightarrow$	(by definition)
$\min(\mathcal{B}, C_n + (T_{n+1} - T_n)/I) \ge 1$	$\Leftrightarrow$	(by (3))
$L + I - Y'_{n+1} \ge I$	$\Leftrightarrow$	(by definition)
the cell is accepted by the conGCRA	$\Leftrightarrow$	(by (2))
the cell is accepted by the GCRA $\blacksquare$		

For $n = 0$, (3) holds by definition. Assuming its correctness for all $0 \le i \le n$, we obtain for a successful event at time T_{n+1}, that $Y_{n+1} = Y'_{n+1} + I$ and

$$\begin{aligned} & I \times \min(\mathcal{B}, (T_{n+2} - T_{n+1})/I + C_{n+1}) \\ &= \ I \times \min(\mathcal{B}, \ (T_{n+2} - T_{n+1})/I + \ \min(\mathcal{B}, C_n + (T_{n+1} - T_n)/I) - 1) \\ &= \ \min(L + I, (T_{n+2} - T_{n+1}) + \ L + I - Y'_{n+1} \ - I), \text{ by induction hypothesis} \\ &= \ L + I - \max(0, Y_{n+1} - (T_{n+2} - T_{n+1})) \ = \ L + I - Y'_{n+2}. \end{aligned} \tag{4}$$

Finally, in the case of cell rejection, we may infer (4) by the same arguments, observing now that $Y_{n+1} = Y'_{n+1}$ and $C_{n+1} = \min(\mathcal{B}, C_n + (T_{n+1} - T_n)/I))$ $\blacksquare$

A.2 The Exponentially Weighted Moving Average Scheme

To derive the credit function for the exponentially weighted moving average scheme (see page 7), we first generalize Eq. (1), such that S_p can be calculated from a previous S_q for any $q < p$:

$$S_p = \alpha^{p-q} S_q + (1 - \alpha) \sum_{k=q+1}^{p} (\alpha^{p-k} X_k) \quad \text{for } 0 \le q < p \tag{5}$$

At a packet event, the current window is given by $w(T_{n+1})$ and consequently, as defined above, the initial credit limit is $S_{w(T_{n+1})-1}$. By rewriting Eq. (5), setting $q = w(T_n)$ and by observing for $w(T_n) < k < w(T_{n+1})$ that $X_k = 0$, we get

$$S_{w(T_{n+1})-1} = \alpha^{w(T_{n+1})-1-w(T_n)} S_{w(T_n)} \quad \text{for } w(T_n) < w(T_{n+1}) \tag{6}$$

Using the auxiliary variables B_n defined in the formula for the credit function, we may, as is proven below, conclude that

$$B_n = S_{w(T_n)} \quad \text{for } w(T_n) < w(T_{n+1}) \tag{7}$$

Combining these results yields the stated formula for the credit function.
To establish Eq. (7), we prove that for the last packet event n in any window we have $B_n = S_{w(T_n)}$: Let $1 = w(T_0) \leq p_1 < p_2 < \ldots < p_k = w(T_n)$ denote the number of those windows, where packet events occurred and $n(p_j)$ the index of the last such event in window p_j, i.e., $n(p_j) = \max\{k \mid w(T_k) = p_j\}$. Then we may deduce

$$\begin{aligned}
B_n = B_{n(p_k)} &= (1-\alpha)X_{p_k} + \alpha^{p_k - p_{k-1}} B_{n(p_{k-1})} \\
&= (1-\alpha)X_{p_k} + \alpha^{p_k - p_{k-1}}((1-\alpha)X_{p_{k-1}} + \alpha^{p_{k-1} - p_{k-2}} B_{n(p_{k-2})} \\
&= (1-\alpha)(X_{p_k} + \alpha^{p_k - p_{k-1}} X_{p_{k-1}}) + \alpha^{p_{k-1} - p_{k-2}} B_{n(p_{k-2})} \\
&\vdots \\
&= (1-\alpha)\sum_{j=1}^{k}(\alpha^{p_k - p_j} X_{p_j}) + \alpha^{p_k - 1} B_0 = (1-\alpha)\sum_{j=1}^{k}(\alpha^{p_k - p_j} X_{p_j}) + \alpha^{p_k}\mathcal{M} \\
&= \alpha^{p_k}\mathcal{M} + (1-\alpha)\sum_{m=1}^{p_k}(\alpha^{p_k - m} X_m), \text{ since } X_m = 0 \text{ for } m \neq p_j, \\
&= S_{p_k}, \quad \text{by equality (5)} \blacksquare
\end{aligned}$$

Acknowledgments

We are indebted to Steve Fuhrman, Ronald Luijten, Doug Dykeman and Bernd Meister for fruitful discussions.

References

[1] E. Sohraby, L. Fratta, I.S. Gopal, and A.A. Lazar, Eds., *Congestion-Control in High-Speed Packet Switched Networks*, vol. 9(7), *IEEE JSAC*, 1991.

[2] A. E. Eckberg, D. T. Luan, and D. M. Lucantoni, "Meeting the challenge: Congestion and flow control strategies for broadband information transport", in *Proceedings of Globecom'89, Dallas, USA*, 1989, pp. 49.3.1–49.3.5.

[3] Maurizio Decina, "Open issues regarding the universal application of ATM for multiplexing and switching in the B-ISDN", in *Proceedings of the ICC'91*, 1991.

[4] K.K. Ramakrishnan and Peter Newman, "Integration of rate and credit schemes for ATM flow control", *IEEE Network Magazine*, pp. 49–56, March/April 1995.

[5] A. E. Eckberg, B. T. Doshi, and R. Zoccolillo, "Controlling congestion in B-ISDN/ATM: Issues and strategies", *IEEE Communications Magazine*, pp. 64–70, Sept. 1991.

[6] B. Leiner, "Critical issues in high-bandwidth networking", RFC 1077, NIC, Nov. 1988.

[7] W. Doeringer, D. Dykeman, M. Kaiserswerth, B. Meister, H. Rudin, and R. Williamson, "A survey of light-weight transport protocols for high-speed networks", *IEEE Transactions on Communications*, vol. 38, no. 11, pp. 2025–2039, November 1990.

[8] J. S. Turner, "A proposed bandwidth management and congestion control scheme for multicast in ATM networks", Tech. Rep., University of Washington, May 1991.

[9] W. Doeringer, D. Dykeman, S. Fuhrmann, M. Heddes, R. Luijten, and B. Meister, "Basic rate-control mechanism for broadband networks", *IBM Technical Disclosure Bulletin*, November 1992.

[10] Marco Heddes, *A Hardware/Software Codesign Strategy for the Implementation of High-Speed Protocols*, PhD thesis, Eindhoven University of Technology, The Netherlands, 1995.

[11] A. Descloux, "Contention probabilities in packet switched networks with strung input processes", in *Proc of the ITC-12, Torino, Italy*, 1988.

[12] S. Fuhrmann and J.Y. Le Boudec, "Burst and cell level models for ATM buffers", in *Proceedings of the ITC-13, Copenhagen, Denmark*, June 1991, pp. 975–980.

[13] ATM Forum, *ATM User Network Interface Specification, Version 3.0*, Prentice-Hall, Englewood Cliffs, New Jersey, first edition, 1993.

[14] M.J. Rider, "Protocols for ATM access networks", in *Proceedings of the IEEE Globecom '88*, 1988, pp. 112–117.

[15] F. Bobomi and K.W. Fendick, "The rate-based flow control framework for the available bit rate ATM service", *IEEE Network Magazine*, pp. 25–39, March/April 1995.

[16] ATM Forum, "Traffic management specification version 4.0", Tech. Rep. atmf95-0013R6, ATM Forum, 1995.

[17] D.D. Clark, M.L. Lambert, and L. Zang, "NETBLT: A bulk data transfer protocol", RFC 998, NIC, 1987.

[18] D. Verma, L. Zhang, and D. Ferrari, "Guaranteeing delay jitter bounds in packet switched networks", in *Proceedings of the TriComm 91, Chapel Hill, USA*, Apr. 1991.

[19] A. Bohn-Nielson, "Resource management", in *Performance Evaluation and Design of Multi-Service Networks*, J.W. Roberts, Ed. Commission of the European Communities, Brussels, Belgium, first edition, 1991, COST 224 TD(90) 019.

[20] K. K. Moth, L. Dittman, and S. Jacobsen, "Flow enforcement algorithms for ATM networks", in *Proceedings of the ISS 90*, 1990.