Flow enforcement algorithms for ATM networks

Dittmann, Lars; Jacobsen, Søren B.; Moth, Klaus

Published in:
I E E E Journal on Selected Areas in Communications

Link to article, DOI:
10.1109/49.76632

Publication date:
1991

Document Version
Publisher's PDF, also known as Version of record

Citation (APA):
Dittmann, L., Jacobsen, S. B., & Moth, K. (1991). Flow enforcement algorithms for ATM networks. I E E E Journal on Selected Areas in Communications, 9(3), 343-350. DOI: 10.1109/49.76632

Flow Enforcement Algorithms for ATM Networks

Lars Dittmann, Søren B. Jacobsen, and Klaus Moth

Abstract—**This paper characterizes four measurement algorithms for flow enforcement in ATM networks. The algorithms are the leaky bucket, the rectangular sliding window, the triangular sliding window, and the exponentially weighted moving average. A comparison, based partly on teletraffic theory and partly on signal processing theory, is carried out. It is suggested to use the RMS measurement bandwidth to dimension linear algorithms for equal flow enforcement characteristics. Implementations are proposed on the block diagram level, and dimensioning examples are carried out when flow enforcing a renewal-type connection using the four algorithms. The corresponding hardware demands are estimated and compared.**

I. Introduction

WITHIN the last few years, ATM has turned out to be one of the most promising concepts for an integrated broadband communication network (IBCN). ATM has the potential to fulfill the demands of flexibility and future robustness required of an IBCN.

However, to be able to benefit from the capabilities that can be obtained by using ATM, it is important that the system aspects of the ATM concept are well understood by the network designers. A very large amount of research has to be carried out to evaluate and compare the different functionalities and thereby establish a sufficient amount of information for the standardization bodies.

Within the last couple of years, a concept referred to as preventive load control [1], [2], [3] has obtained increasing interest as a resource allocation strategy. The objective of preventive load control is to avoid congestion by a restrictive and conservative access control. Preventive load control applies two key functions: admission control and flow enforcement (policing).

The work concerning admission control is mainly related to the ATM switch nodes, where traffic from a number of incoming connections interferes. The admission control (AC) scheme must ensure that the quality of service (QOS) level (e.g., cell loss rate, delay, and delay jitter) inside the switch node is kept at an acceptable level. For each new connection request, the AC scheme maps a set of parameters for the new connection and a corresponding set for the established connections into a QOS estimate. The QOS estimate is used to decide whether or not the new connection request can be accepted. The AC parameters are declared at connection setup and the traffic must adapt to these bounds until the connection is released, or a renegotiation is initiated. In preventive load control, it is mandatory to supervise each connection at the entrance to the network in order to make the QOS estimate reliable. If a violation of the agreement (either caused by a malfunctioning terminal or due to a deliberate attempt) is detected, cells will be dropped or, as suggested in [4], marked so that they can be dropped in a later stage.

The function that is supervising the connection has been given many names in literature, such as policing, flow enforcement, traffic enforcement, and usage parameter control (the last is used by CCITT, see [5]). In this paper, we have chosen the name flow enforcement because we fell it gives a better figurative description of the function (and, furthermore, has been adopted within the RACE 1014 project, "ATMOSPHERIC", where many of the ideas presented in this paper first arose).

Requirements of the parameters used to characterize a connection must meet at least two important constraints.

1) They must be able to accurately describe the traffic characteristics of the connection, such that the QOS estimate is reliable.

2) They must be easy to supervise to make the implementation of the flow enforcement algorithm feasible and cost-effective.

While the first demand has been given much attention, the second is rarely covered in present literature.

In order to obtain a flexible and future proof network, several requirements are associated with the flow enforcement function [6].

1) The flow enforcement function should be service independent. It will only use different parameter settings for different services.

2) The flow enforcement function must supervise each connection in order to have the possibility to point out the actual connecting violating its obligations.

3) The flow enforcement function must be cost effective, which implies that it must be simple to implement.

4) A terminal should have the possibility to emulate the actual flow enforcement function within the terminal in order to achieve good utilization of allocated resources and to prevent the flow enforcement function from discarding any cells.

5) The requirement 4) and the possibility to move a terminal from one customer premises network (CPN) to another demands a standardization of the flow enforcement function supervising traffic from the CPN.

Recently, a number of papers have questioned the feasibility of flow enforcement, see [7] and [8]. A main argument against flow enforcement is that, in order to flow enforce a statistical source close to its mean rate, a large time constant is needed to avoid excessive cell loss, and the large time constant implies inefficient control of the traffic flow. However, by employing a flow throttling function in the sending terminal (or elsewhere in the CPN) [3], it is possible to flow enforce a source close to its

Manuscript received April 20, 1990; revised November 21, 1990. This work was supported by the RACE "ATMOSPHERIC" Project R1014 and the Centre for Broadband Telecommunication, Electromagnetics Institute, Technical University of Denmark, Lyngby, Denmark. This paper was presented at the R1012 Broadband Local Network Technology (BLNT) RACE Workshop on Traffic and Performance Aspects in IBCN, Munich, Germany, July 3-4, 1990.

L. Dittmann is with the Electromagnetics Institute, Technical University of Denmark, DK 2800 Lyngby, Denmark.

S. B. Jacobsen is with the ELLEMTEL Telecommunication System Laboratories, IDEON, S-22370 Lund, Sweden.

K. Moth is with Alkatel Kirk, Aerospace Systems Division, Lautrupvang 2, DK 2750 Ballerup, Denmark.

IEEE Log Number 9042038.

mean bit rate, even with a relatively small time constant. The flow throttler is used to shape the traffic in accordance with the negotiated parameters. This is done by emulating the FE function. But, instead of discarding cells, the flow throttler uses a buffer to delay violating cells. By ensuring that the mean bit rate is below its limit within the shorter time period, it is also ensured that the requirement is fulfilled for the longer period. Furthermore, by shaping the source behavior, it is possible for the user to utilize the required capacity to a higher extent and thereby make the connection more cost effective. However, the use of a traffic shaper strongly requires a standardization of the FE function.

If a burst allocation protocol like the one described in [9] is implemented, requirements of services with long burst durations can also be met by using peak allocation during the burst period. The issue will not be addressed further in this paper.

A comparison between four different flow enforcement algorithms, based primarily on teletraffic analysis, has been carried out in [10]. An alternative criterion for comparison based on an efficiency parameter can be found in [11]. The objective of this paper is to compare and evaluate four different flow enforcement algorithms. The algorithms are the leaky bucket (LB), the rectangular sliding window (RSW), the triangular sliding window (TSW), and the exponentially weighted moving average (EWMA). The basis for the comparison is a mix of teletraffic and signal processing theory.

Traditional teletraffic analysis makes it possible to calculate the performance of the leaky bucket and the rectangular sliding window (at least for the Poisson arrival process). In the two other cases, no analytical tool seems available and simulations are necessary. However, both these algorithms and the RSW are linear algorithms and the traditional signal processing theory applies. Equivalents are obtained by means of the Z-transform of the transfer functions for the different algorithms. This approach is evaluated through simulations for different traffic types.

In Section II, the different algorithms are presented and the Z-transforms are calculated for the transfer functions in the linear cases. Section III contains a description of how the algorithms can be equivalated, either by teletraffic tools or by means of linear signal processing. Evaluation and implementation suggestions are given in Section IV, including numerical examples.

II. Flow Enforcement Algorithms

A flow enforcement (FE) algorithm can be divided into a measuring algorithm that provides the FE parameters and an action to be taken when certain constraints are fulfilled by these parameters (e.g., a certain threshold is reached). The measuring algorithms monitor connections and does not affect the cell streams; it involves some signal processing, such as averaging. The action algorithm interferes if certain threshold values are violated.

Certain requirements to FE must be fulfilled.

1) The action to be taken when a certain connection violates its FE parameters must concern the actual cell that causes the violation.

2) The FE algorithm can only use measurement data that has been provided before the arrival of the cell in question. For example, no cell delay (that would include the following behavior in the decision) should be allowed.

One of the well-known FE algorithms is the leaky bucket [12]. The algorithm works as follows. A counter value is decreased periodically by a fixed amount D until the value zero has been reached. When the counter value reaches zero, the decrease rate is changed to zero. For each accepted cell arrival from the considered connection, the counter value is increased by an amount I.

The parameters are related as

$$f_{\text{LB}} = \frac{D}{I} f_{\text{link}} \tag{2.1}$$

where f_{LB} is the long-term allowed bandwidth and f_{link} is the bandwidth of the link. Either D or I may be fixed for different connections in a realization.

The action algorithm is invoked when the counter value exceeds a given threshold value C_T.

The functionality of the leaky bucket is similar to the access class mechanism in the Switch Multimegabit Digital Service, the only difference being that the SMDS mechanism must take into account variable packet length, see [13].

The leaky bucket algorithm is an example of a nonlinear measuring algorithm.

A measuring (averaging) algorithm A, applied to the signal s_1 and s_2, is called linear and time invariant if:

1) $A(as_1 + bs_2) = aA(s_1) + bA(s_2)$

2) $A(s_1(n-k)) = As_1(n-k)$.

A linear time-invariant algorithm is uniquely determined by its unit sample response h, see [14], in this context called the transfer function.

The three FE algorithms RSW, TSW, and EWMA are all linear and time invariant.

Rectangular Sliding Window (RSW)

The cell arrivals within a time window of length N are monitored. The transfer function is independent of j (j being the position in the window) and is given by

$$h_{\text{RSW}}(j) = \frac{1}{N} \quad \text{if } j = 0, 1, 2, \cdots, N-1, \text{ else } 0. \tag{2.2}$$

The power spectrum (by z-transform) may be written as

$$\left|H^2(f)\right| = \frac{\sin^2(N\pi f\Delta t)}{(N^2 \sin^2(\pi f \Delta t))} \tag{2.3}$$

where Δt is the time required to transmit an ATM cell on a link ($=2.7\ \mu$s on a 155.52 Mb/s link).

Triangular Sliding Window (TSW)

This weight function is given by

$$h_{\text{STW}}(j) = \frac{j}{\alpha} \quad \text{if } j = 1, 2, 3, \cdots, N, \text{ else } 0 \tag{2.4}$$

where $\alpha = \Sigma_1^N j = [N(N+1)/2]$ is a normalization factor.

The power spectrum is given by

$$\left|H^2(f)\right| = \frac{\alpha^2\Big((N^2+1+N) - (N+1)\big(N\cos(2\pi f\Delta t) + \cos((N+1)2\pi f N \Delta t)\big)\Big)}{3 + \cos(4\pi f\Delta t) - 4\cos(2\pi f \Delta t)}. \tag{2.5}$$

Exponentially Weighted Moving Average (EWMA)

The general EWMA algorithm can be written as a function p that, at the discrete-time $n\Delta t$, has the value $p(n\Delta t)$, where x is a binary function that has the value $x(k\Delta t) = 1$ in case of cell arrival and otherwise has the value $x(k\Delta t) = 0$.

$$p(n) = \phi \sum_{i=0}^{N} x((n - N + j)\,\Delta t)\,(1 - \mu)^{N-j} \quad (2.6)$$

where $\phi = [\mu/(1 - (1 - \mu)^N)]$ is the normalization constant. Thus,

$$h_{\mathrm{EWMA}}(j) = \phi(1 - \mu)^j \quad \text{if } j = 0, 1, 2, 3, \cdots, N, \text{ else } 0. \quad (2.7)$$

When N is an infinite number of samples, $\phi = \mu$.

Equation (2.6) may be written as a recursive expression:

$$p(n\Delta t) = (1 - \mu)p((n - 1)\Delta t) + \phi x(n\Delta t). \quad (2.8)$$

Thus, initially, the window length is infinite due to the recursive expression. The power spectrum of the weight function may be written [14] as

$$|H^2(f)| = \frac{\mu^2}{(1 + (1 - \mu)^2 - 2(1 - \mu)\cos(2\pi f \Delta t))}. \quad (2.9)$$

TSW and EWMA are similar in the way that the newest samples are given the highest weight. It should be denoted that the definition, used in this paper, of the EWMA algorithm differs from the definition given in [10] in the sense that the EWMA defined in [10] only makes one update for each window length.

III. Characterization and Comparison of FE Algorithms

This section discusses how to stipulate the real parameters for dimensioning an algorithm when a connection is going to be flow enforced. A common basis of comparison of flow enforcement algorithms is only provided when the same connection, with specific traffic behavior, is flow enforced by the different FE algorithms.

A comparison of the measurement algorithms is carried out on this basis. Criteria are estimates on circuit complexity and suitability to different ranges of parameters. The comparison is made on the basis of a fixed cell loss probability for all the algorithms. However, not only the cell loss probability might be of interest when selecting a flow enforcement algorithm. For instance, both the TSW and EWMA algorithms give a higher weight to newly arrived cells, which gives a better opportunity to react on sudden changes in source behavior. (This topic, however, will not be addressed further in this paper).

FE Parameters versus Dimensioning of FE Algorithms

It is possible to estimate the FE parameters from the source behavior using several techniques. A model of the source behavior and a model of the algorithm is required. The *source* may be described by a well-known arrival process, e.g., a renewal process. In order to limit the cell loss in the FE and to achieve a reasonable time constant (window length), the source must be enforced according to a rate $\lambda_{\mathrm{FE}} > \lambda_s$ (λ_s being the mean arrival rate of the source).

The *algorithm* may be described by an equivalence to a specific queueing or multiserver models.

Linear Averaging: In the case of the rectangular sliding window, there is an equivalence to a multiserver loss system [15] with a deterministic service time equal to the length of the window (N), i.e., the service rate is $1/N$. If the source is Poissonian, the probability distribution of the number of occupied servers (number of cells from the source in the window) is independent of the distribution of the service time and the classical Erlang-B formula can be used to calculate the cell loss ratio.

There is no immediate equivalence in the case of TSW or EWMA. However, as both algorithms together with the RSW are performing a linear averaging, they can be equivalated with a filter and the signal theory can be applied through the use of the Z-transformation and an equivalent bandwidth.

Calculation of the equivalent bandwidth is based on the Z-transform of the transfer function $H(f)$ or the power spectrum $H^2(f)$. In this paper, we use the RMS bandwidth for comparison. This equivalence is commonly used to compare the information bandwidth of different types of electrical filters, and is similar to the comparison we want to carry out for the flow enforcement algorithms, where we want to set up a dimensioning criteria, based on cell loss probability, for algorithms that we cannot examine by means of teletraffic theory.

The RMS bandwidth B_e of a power spectrum $H^2(f)$ used as a bandwidth measure is defined as:

$$B_e^2 = \frac{\int_{f=0}^{1/2\Delta t} f^2 H^2(f)\,df}{\int_{f=0}^{1/2\Delta t} H^2(f)\,df} \quad (3.1)$$

The procedure is now:

- specify a real window length N_{RSW} by using multiserver equivalence (see previous section),
- calculate equivalent bandwidth $B_{e,\mathrm{RSW}}(N_{\mathrm{RSW}})$ using (3.1),
- find the parameters μ for EWMA and N_{TSW} for the TSW by using (3.1) to set

$$B_{e,\mathrm{EWMA}}(\mu) = B_{e,\mathrm{RSW}}(N_{\mathrm{RSW}}) \quad \text{and} \quad B_{e,\mathrm{TSW}}(N_{\mathrm{TSW}}) = B_{e,\mathrm{RSW}}(N_{\mathrm{RSW}}).$$

The parameters N_{RSW}, $\mu(B_{e,\mathrm{RSW}})$ and N_{TSW} may be used for dimensioning for FE implementation. A numerical example with three different types of renewal traffic are given in Section IV. The results obtained by simulations are used to verify the filter equivalent for linear averaging algorithms.

The *leaky bucket* may be modeled using the equivalence between the LB and a queue with constant size (I/D) batch arrivals, and deterministic service time ($G^x/D/1$ queue). If the source is described by a Poisson process, the classical $M^x/D/1$ batch queue may be applied. The load on the queue is given by $\rho = \lambda_s/\lambda_{\mathrm{LB}} = (I/D)(\lambda_s/f_{\mathrm{LB}})$, where λ_s is the arrival intensity and λ_{LB} is the parameter to be flow enforced. The model calculates the cell loss ratio ϵ (when applying the load ρ to the queue). This cell loss ratio will be specified not to exceed a certain limit, and a minimum required buffer length Q can be derived. The threshold value is related to the decrement value by: $C_T = QD$.

Measurement Quantization

Each measurement algorithm is accomplished with a certain quantization. Thus, a measured parameter, e.g., estimated mean, cannot be represented more accurately than corresponding to this uncertainty. One example is the RSW: a window of length of 2 implies that it is only possible to distinguish between bandwidths exceeding 50% of a link bandwidth or not exceeding this value. A larger window would improve quantization but the averaging time would increase, resulting in a less representative peak value.

The quantization may vary for different values of the measured parameters. Thus, only a worst-case figure $\Delta p_{\max} = \max\{p_k - p_{k-1}\}$ is derived, defined as the maximum difference between two measurable values (p_k, p_{k-1}) next to each other (e.g., p being a measurable mean bit rate).

The quantization is stipulated by the measurement window shape, the window length, and internal representation of function values.

The objective is to have a linear and small quantization in order to provide the largest flexibility with respect to the bandwidth parameter. Otherwise, the flexibility of ATM cannot be fully exploited. However, it may be acceptable to allow for larger granularity when specifying large bandwidths than in the case of low bandwidth channels.

Leaky Bucket: In this case, the worst-case quantization is given by the internal representation of the counter value (see [6], Chap. 2):

$$\Delta p_{\max} = 1/I_0 \quad \text{(fixed-increment } I_0\text{, lin. quantization)} \tag{3.2}$$

or

$$\Delta p_{\max} = 1/(1 + D_0) \quad \text{(fixed-decrement } D_0\text{, non-lin. quantization)}. \tag{3.3}$$

Linear Averaging: The sample space of these functions is, in general, found by a permutation of any combination of 1's and 0's inside the window.

An exception is EWMA, where the granularity of the function value is given by $(2^b - 1)^{-1}$, where b is the number of bits used to represent the function value.

IV. Implementation Aspects

This section deals with actual proposals for implementation. The complexity is commented upon. The objective of the discussion is that FE, for cost reasons, should be highly integrated and thus as hardware-oriented as possible. It is recognized that most algorithms may be implemented by complex computer circuitry. However, optimum integration demands for dedicated design.

A. Proposals for Implementation

Periodical updating or memory access every Δt, equal to the cell duration, for each connection on a physical ATM line imposes strict speed constraints. Equation (4.1) yields the maximum allowable time τ for carrying out the algorithm update:

$$\tau = \frac{\Delta t}{N_L N_{\text{VC}}} \tag{4.1}$$

where N_L is the number of links handled by the FE hardware, N_{VC} is the number of individual virtual channels. A link speed of 155.520 Mb/s, one link, 12 bits connection identifier, and a cell size of 53 octets yields $\tau = 0.7$ ns. This is close to the limit of what is possible for small (1K addresses) state-of-the-art memory circuits [16], [17].

TABLE I
Measurement Quantization of Four FE Algorithms

	$\Delta p_{\max}$	Behavior	Type
RSW	$1/N$	const.	window
TSW	$\alpha = 2/(N(N+1))$	const.	window
EWMA	$(2^b - 1)^{-1}$	const.	internal
Leaky Bucket (fixed D_0)	$1/(D_0 + 1)$	var.	internal
Leaky Bucket (fixed I_0)	$1/I_0$	const.	internal

Legend:
Internal: caused by internal representation of function values.
Window: due to window length and shape.
Const.: constant Δp through the sample space.
Var.: varying Δp through the sample space.

For large $N_L N_{\text{VC}}$, timediveded application of the FE algorithm is necessary. This can be achieved by the use of time stamps that keeps information on the previous cell arrival.

In this way, extra hardware is required to handle and store time stamps, but the time for each update will be independent of N_{VC}:

$$\tau = \frac{\Delta t}{N_L}. \tag{4.2}$$

Furthermore, the output of the averaging function value of a certain connection is only of interest when a cell from that specific connection arrives. Using the previous example yields $\tau = 2.7$ μs, which is met by most semiconductor technologies for even large memories [16].

Timing aspects are not considered in the following sections. However, notice that the FE action (accept, discard, or mark cell) should be provided as soon as possible in order not to delay the cell unnecessarily.

Leaky Bucket

In a time stamp implementation of the LB algorithm, the periodic decrementing of the counter value of each channel is reduced to a single subtraction of every cell arrival. The value to subtract is the decrement value times the distance to previous cell arrival of that specific connection.

The algorithm may be described (case: fixed increment) by pseudocode:

```
For each cell arrival do:
$\Delta n := n_{\text{last}}(\text{VCI}) - n_{\text{now}}$          {timestamp handling}
$n_{\text{last}} := n_{\text{now}}$
$C := C(\text{VCI})$
if $C < \Delta n D(\text{VCI})$                    {bucket decrement}
    then $C := 0$
    else $C := C - \Delta n D(\text{VCI})$
if $C > C_T(\text{VCI}) - I_0$   {this value is in store}
  {FE action}
    then discard_cell
    else $C(\text{VCI}) := C + I_0$             {bucket increment}
```

Fig. 1 outlines a proposal for implementation using fixed increment. The counter memory CM contains the "bucket" value of each connection, C and the time stamp, n_{last}. The FEP memory contains the FE parameters of each connection: D and $C_T(-I_0)$. The complexity is outlined by Table II. $\{x\}$ denotes the number of bits necessary to represent the number x.

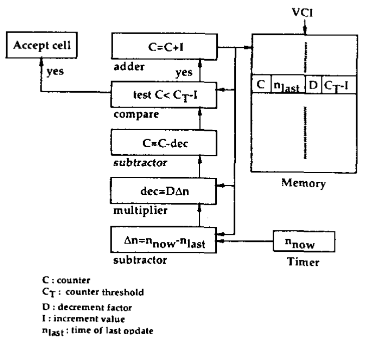


Fig. 1. Implementation proposal for the LB algorithm (with fixed increment value).

TABLE II
COMPLEXITY OF LEAKY BUCKET IMPLEMENTATION (FIXED INCREMENT) EXPRESSED BY THE SIZE OF KEY COMPONENTS. S IS THE NUMBER OF BITS USED TO REPRESENT TIME STAMPS

Component	Length	Width	Comment
Counter Memory	N_{VC}	$\{C_T\} + S$	
FEP Memory	N_{VC}	$\{C_T - I\} + \{D\}$	
Adder (decrement)	—	$\{C_T\}$	per inlet
Adder (increment)	—	$\{C_T\}$	per inlet
Adder (timestamp)	—	S	per inlet
Multiplier	—	$\max(S, \{D\})$	per inlet
Comparator	—	$\{C_T\}$	per inlet

The leaky bucket is flexible to changes in FE constraints: all parameters may be varied within the range allowed by their representation.

Sliding Windows

In general, a sliding window of length N can be implemented using a shift register containing N channel identifiers. Every Δt the shift register is updated: the new cell identifier is shifted into the top of the register and the bottom cell identifier is shifted out. If the weight function is nonconstant, there must be access to the information of each position of the window (which excludes a normal FIFO structure).

There are two basic ways of implementing the weight function.

Case 1) Each connection has its own shift register and circuitry for addition of weight function values. Only the values 1 or 0 are shifted into the register. At each cell arrival, all registers are shifted: 0's into all registers except for the register with the identity equal to the cell identifier.

The computation could be very simple as all possible sums (the sample space) could be contained in a memory lookup table using the shift register contents as lookup address.

Case 2) The shift register could be common to all connections. Each location contains the full connection identifier. Each shift register has its own circuitry for addition or subtraction of its weight function value. The sum to operate on is determined by the shift register contents. When a certain connection identifier enters a location in the shift register, the sum of that connection is updated by the actual weight function value. At the next shift (after Δt), this weight function value is subtracted again. (This corresponds to deleting the sum after each Δt and computing a new sum based on new weight function entries.) However, this implies that only one window size is possible for all connections.

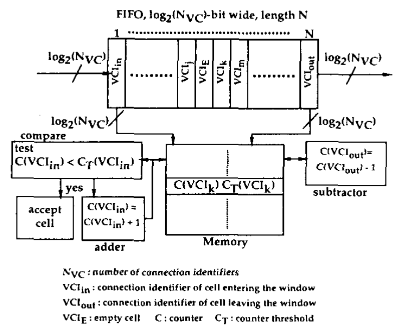

FIFO, $\log_2(N_{VC})$-bit wide, length N
1
N
$\log_2(N_{VC})$
VCI_{in}
VCI_j
VCI_E
VCI_k
VCI_m
VCI_{out}
$\log_2(N_{VC})$
$\log_2(N_{VC})$
$\log_2(N_{VC})$
compare
test
$C(VCI_{in}) < C_T(VCI_{in})$
yes
accept cell
$C(VCI_{in}) = C(VCI_{in}) + 1$
adder
$C(VCI_k)$ $C_T(VCI_k)$
Memory
$C(VCI_{out}) = C(VCI_{out}) - 1$
subtractor
N_{VC} : number of connection identifiers
VCI_{in} : connection identifier of cell entering the window
VCI_{out} : connection identifier of cell leaving the window
VCI_E : empty cell
C : counter
C_T : counter threshold

Fig. 2. Implementation proposal for the RSW algorithm.

A basic assumption in this paper is that an implementation of a flow enforcement algorithm should be able to supervise a large number (several hundreds) of connections in order to be cost effective. The requirement of a cost-effective implementation excludes the proposal outlined in case 1 and, therefore, only case 2 will be covered in the following.

Rectangular Sliding Window

In this case, the implementation can be very simple, as actions only have to be carried out when a connection identifier enters or leaves the window. Fig. 2 outlines the function of the RSW. The FIFO contains the connection identifiers of cells in the window. When a cell connection identifier enters the window, the counter value of this connection is increased by one. This is done by providing the counter value from the counter memory (CM), incrementing it, and returing it to the CM. Correspondingly, when a cell identifier leaves the FIFO, its counter value is decremented.

Fig. 2 outlines the proposed implementation. The complexity requirements are outlined by Table III. The maximum window length is stipulated by the FIFO length. Thus, the RSW is not flexible to any changes in FE constraints.

Triangular Sliding Window

Again, the general way of implementation can be avoided by using the technique outlined by Fig. 3. It is exploited that the same operation is carried out on the counter value for every Δt as long as the cell (identifier) is present in the window. Thus, one part on the circuit manages the connections presence in the window, the other part updates the function value. A memory is used to store the number of cells in the window of a specific connection $k(\text{VCI})$ and a timestamp n_{last} that indicates the last time the counter value $C(\text{VCI})$ of that specific connection was updated.

This operation is carried out for every cell arrival (connection identifier: VCI) (in pseudocode).

```
For every cell arrival do:
$\Delta n := n_{\text{now}} - n_{\text{last}}(\text{VCI})$   {computes distance to last update}
$n_{\text{last}}(\text{VCI}) := n_{\text{now}}$ {update time stamp}
$C(\text{VCI}) := C(\text{VCI}) - k(\text{VCI})\Delta n$
```

TABLE III
COMPLEXITY OF KEY COMPONENTS IN RSW IMPLEMENTATION

Component	Length	Width	Comments
FIFO	N	$\{N_{VC}\}$	—
Counter Memory	N_{VC}	$\{N\}$	—
FEP Memory	N_{VC}	$\{N\}$	—

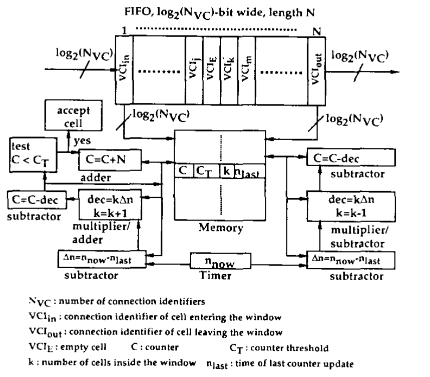


Fig. 3. Implementation proposal for the TSW algorithm.

```
if $C(\mathrm{VCI}) + N > C_T(\mathrm{VCI})$ then discard_cell    {FE action}
    write "empty cell" into FIFO
else accept_cell
    $C(\mathrm{VCI}) := C(\mathrm{VCI}) + N$    {increase countervalue}
    $k(\mathrm{VCI}) := k(\mathrm{VCI}) + 1$    {increase number in window by one}
    write VCI into FIFO
endif
```

```
For every cell that leaves the FIFO:
$\Delta n := n_{\mathrm{now}} - n_{\mathrm{last}}(\mathrm{VCI})$    {computes distance to last update}
$C(\mathrm{VCI}) := C(\mathrm{VCI}) - k(\mathrm{VCI})\Delta n$
$k(\mathrm{VCI}) := k(\mathrm{VCI}) - 1$    {reduce number in window by one}
$n_{\mathrm{last}}(\mathrm{VCI}) := n_{\mathrm{now}}$
```

Time stamps n_{last} are used to avoid that all C-values need to be decreased simultaneously.

A FIFO register manages the number of cells present in the window in order to be able to calculate the decrement value. The implementation requirements are given in Table IV.

Due to the FIFO managing the cells in the window, it is not possible to change the window length after the TSW has been implemented.

EWMA

The time setup techniques are not suited for this algorithm, as the action to compensate for missing updates is rather complex: namely to apply a coefficient $(1 - \mu)^x$ if x slots have passed since the last update. This will require an implementation of a power function to work on non-integer values. As x might be very large, this calculation will be very time consuming and very difficult to carry out in real time, i.e., within the cell duration time.

TABLE IV
COMPLEXITY OF TSW IMPLEMENTATION OUTLINED BY SIZE OF KEY COMPONENTS. S IS THE NUMBER OF BITS USED TO REPRESENT TIME STAMPS

Component	Length	Width	Comments
Counter Memory	N_{VC}	$\{N(N+1)/2\} + S$	
FEP memory	N_{VC}	$\{N(N + 1/2\}$	
Counter Adder	—	$\{N(N+1)/2\}$	per inlet
Time stamp Adder	—	S	per inlet
Multiplier	—	$\{N\}$	per inlet
FIFO	N	$\{N_{VC}\}$	

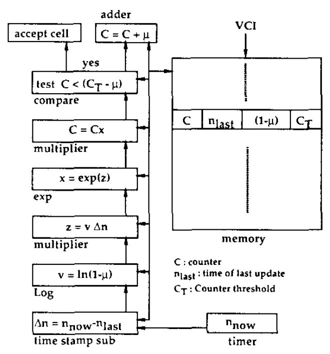


Fig. 4. Implementation proposal for the EWMA algorithm.

```
For every cell arrival do
$C := C(\mathrm{VCI})$    {read counter value}
$\Delta n := n_{\mathrm{now}} - n_{\mathrm{last}}(\mathrm{VCI})$    {time stamp handling}
$x := \exp\,[(\Delta n \log(y(\mathrm{VCI}))]$    {$y = 1 - \mu$}
$C := xC$    {apply coefficient $\Delta n$ times}
$n_{\mathrm{last}}(\mathrm{VCI}) := n_{\mathrm{now}}$
if $(C + (1 - y(\mathrm{VCI}))) > C_T(\mathrm{VCI})$    {test against FEP}
    then discard_cell
    else $C(\mathrm{VCI}) := C + (1 - y)$    {store counter value}
```

Fig. 4 outlines a proposal of implementation, and Table V: outlines the implementation complexity. Notice that synchronous implementation of the EWMA requires rather complex logarithmic and exponential functions. The EWMA gives full flexibility with respect to FE constraints. For example, μ and, thus, the window length can be truncated to the value on selects within the range of the number representation.

B. Example of Dimensioning FE Algorithms

This section gives an example of dimensioning the different algorithms based on flow enforcement of three different renewal processes, an Erlang-5, a Poisson, and hyperexponential with squared coefficient of variation equal to 5. Thereby, both smooth and bursty traffic have been modeled. Furthermore, techniques have been derived for approximating more general point processes by renewal processes, see [18], thus, in principle, extending the generality of the example to cover correlated traffic. The three examples are used to evaluate the filter equivalence by the RMS method for both smooth and bursty traffic.

We consider a source with a mean bit rate of $\lambda_s = 10$ Mb/s and an enforced rate of $\lambda_{\mathrm{FE}} = 16$ Mb/s. The maximum cell loss probability is chosen to be $\epsilon = 10^{-4}$. The quantization is

TABLE V
IMPLEMENTATION COMPLEXITY OF EWMA ALGORITHM CONSIDERING KEY COMPONENTS. $b = \mathrm{LOG}_2(\delta)$. S IS THE NUMBER OF BITS USED FOR TIME STAMP REPRESENTATION

Component	Length	Width	Comment
Counter Memory	N_{VC}	$2b + S$	
FEP memory	N_{VC}	b	
Time stamp subtr.		S	inlets
Adder		b	inlets
Log. function		b	inlet
Exp. function		b	inlet
Multiplier ($\Delta n \cdot \log(y)$)		b	inlets
Multiplier		b	inlets

TABLE VI
SIMULATION RESULTS COMPARED TO RESULTS OBTAINED BY USING THE FILTER EQUIVALENT

		Poisson	Erlang-5	Hyperexp.
RSW	(Analytical model)	$N = 540$ $C_T = 64$		
RSW	(Simulation)	$N = 540$ $C_T = 64$	$N = 110$ $C_T = 13$	$N = 632$ $C_T = 75$
TSW	(RMS equivalent)	$N = 810$ $C_T = 48$	$N = 165$ $C_T = 9.7$	$N = 948$ $C_T = 56.2$
TSW	(Simulation)	$N = 777$ $C_T = 46$	$N = 186$ $C_T = 11$	$N = 946$ $C_T = 56$
EWMA	(RMS equivalent)	$\mu = 0.0019$ $C_T = 0.1$	$\mu = 0.0093$ $C_T = 0.1$	$\mu = 0.0017$ $C_T = 0.1$
EWMA	(Simulation)	$\mu = 0.0019$ $C_T = 0.11$	$\mu = 0.0093$ $C_T = 0.11$	$\mu = 0.0017$ $C_T = 0.11$
LB	(Simulation)	$I/D = 8.44$ $C_T = 9.6$	$I/D = 8.44$ $C_T = 2.8$	$I/D = 8.44$ $C_T = 11.9$

chosen to be: $\Delta p_{\max} = 16$ kb/s/135 Mb/s. The example concerns an ATM link of $\lambda_{\text{link}} = 135$ Mb/s payload.

The equivalent multi server loss system ($M/M/n/\text{loss}$) is used for the RSW as a reference. Input rate from the source is $\lambda_s = 10$ Mb/s/135 Mb/s $= 1/13.5$, and service rate of the system is $s = 1/N$ (N being the window length).

Flow enforcing at $\lambda_{\text{FE}} = 16$ Mb/s implies that $n = N\lambda_{\text{FE}}/\lambda_{\text{link}}$ cells are allowed in the window. A cell loss ratio of $\epsilon = 10^{-4}$ is required. This yields $n = 64$, corresponding to a window length of $N = 540$. The equivalent window length becomes (from 3.1) $N_{e,\text{RSW}} = 87.69$. This results in $N_{\text{TSW}} = 810$ and for EWMA $\mu = 1.9 \cdot 10^{-3}$.

Table VI shows the results from the simulation used to verify the filter equivalence. All the results are related to the same cell loss probability, $\epsilon = 10^{-4}$. As the LB is nonlinear, no filter equivalent exists and only simulation results are given (to be used in the following for comparison of hardware requirements).

From Table VI, it can be seen how different types of sources change the requirement to the FE algorithm. The results also show an agreement between the simulation results and the RMS filter equivalent.

Hardware Comparison: The comparison concerns the amount of memory required to implement the different FE algorithms, based on the results from the previous example in the case with a Poisson source model. As memory is one of the most hardware demanding parts in the implementation of an FE algorithm flow enforcing several connections, this is a fair indicator of the general hardware requirement and therefore is useful for comparison.

For the LB, the increment factor is given by the required granularity (from Table 1): $I_o = 135$ Mb/s/16 kb/s $= 8437.5$. From the examples, this yield: $I_o = 16875$ (15 b), $D = 2000$ (11 b), and $C_T = 162000$ (18 b), this corresponds to a burst of about 20 cells. From Table II, this results in a total memory comsumption of 60 b.

TABLE VII
MEMORY LOCATIONS PER CONNECTION OF DIFFERENT FE ALGORITHMS WHEN ENFORCING A POISSON SOURCE WITH MEAN BIT RATE 10 Mb/s ON A 135 Mb/s ATM LINK. N_{VC} IS 4096, CORRESPONDING TO 12 BITS FOR CONNECTION IDENTIFICATION. 16 BITS ARE USED FOR TIME STAMP REPRESENTATION

Algorithm	Memory Locations Per Connection	FIFO Dimensions	Comments
Leaky Bucket	60	—	—
Rectangular SW	20	540×12	no time stamps
Triangular SW	54	810×12	—
EWMA	58	—	—

For EWMA, the required amount of memory is stipulated by the representation μ (11 b), and the requirement with respect to quantization, the last term being the most strict ($b = 14$ b). Table V is used to estimate the total consumption of memory for the EWMA.

Table VII lists the number of memory locations required to implement the different algorithms based on the example in the case with a Poisson source. The figures for the RSW and TSW are derived using Table III and IV.

It can be seen from Table VII that, in the example, the RSW is significantly less demanding with respect to hardware complexity than the other algorithms, a conclusion in strict contrast to those given in [10]. However, their proposal for implementation seems inappropriate.

In the present example, where it is possible to flow enforce 4096 connections simultaneously, the hardware used for FIFO implementation is relatively small. However, if connections down to 16 kb/s (as chosen for the granularity) are to be supervised by an RSW or an TSW, a window length of at least 8438 is required, which corresponds to the ideal cell spacing. Although it is easier to construct a large FIFO memory than RAM storage with a wide address area, a comparison based solely on memory locations are no longer fair if FIFO's of that depth are to be used.

As the LB and EWMA algorithms do not require a buffer, these implementations are less sensitive to the connection bitrate.

V. Conclusion

In this paper, four different flow enforcement (usage parameter control) algorithms have been presented and characterized. The algorithms have been compared with respect to the traditional teletraffic approach, where it is seen that the time constant involved increases with the increasing burstiness of the connection. As a new approach, it has been suggested to use the RMS measurement bandwidth to dimension linear measuring algorithms, an approach which is supported by simulation results. However, we have not been able to verify the approach theoretically, and whether or not this is possible is an open question.

Implementation aspects of the algorithms are discussed at the block diagram level, and a comparison of their hardware demand is carried out. It is concluded that the rectangular sliding window, in contrast to most belief, is competitive with respect to cost-effective implementation. However, its lack of flexibility (the window length cannot be changed after implementation) is a major drawback, a drawback which the leaky bucket and the EWMA algorithm are not burdened with.

Acknowledgment

The authors would like to thank the reviewers for valuable comments.

References

[1] G. M. Woodruff, R. G. H. Rogers, and P. S. Richards, "A congestion control framework for high-speed integrated packetized transport," in *Proc. GLOBECOM '88*, Florida, 1988, pp. 203-207.

[2] S. B. Jacobsen, K. Moth, L. Dittmann, and K. Sällberg, "Load control in ATM networks," in *Proc. ISS'90*, Stockholm, Sweden, 1990, paper A8.5.

[3] K. Sällberg, B. Stavenow, and I. Andersen, "A resource allocation framework for B-ISDN," in *Proc. ISS'90*, Stockholm, Sweden, 1990, paper A2.4.

[4] A. Eckberg, D. T. Luan, and D. M. Lucantoni, "Meeting the challenge: Congestion and flow control strategies for broadband information transport," presented at *GLOBECOM'89*, Dallas, TX, 1989.

[5] CCITT Study Group XVIII, Draft Recommendation I.311.

[6] K. Moth, S. B. Jacobsen, L. Dittmann, K. Sällberg, and K. Venieris, "Final recommendations on management of network resources," RACE "ATMOSPHERIC" project R1014 deliverable E.3.2.B, Dec. 1988.

[7] K. Tutufor, "On admission control and policing in ATM-based networks," in *Proc. 7th ITC Special. Sem.*, Morristown, NJ, 1990, session 5.4.

[8] E. P. Rathgeb and T. H. Theimer, "The policing function in ATM networks," in *Proc. ISS '90*, Stockholm, Sweden, 1990, session A8.4.

[9] P. Boyer, "A congestion control for the ATM," in *Proc. 7th ITC Special. Sem.*, Morristown, NJ, 1990, session 4.3.

[10] E. P. Rathgeb, "Policing mechanisms for ATM networks—Modelling and performance comparison," in *Proc. 7th ITC Special. Sem.*, Morristown, NJ, 1990, session 10.1.

[11] F. Borgonovo and L. Fratta, "Policing in ATM-networks: An alternative approach," in *Proc. 7th ITC Special. Sem.*, Morristown, NJ, 1990, session 10.2.

[12] J. Turner, "New directions in communications (for Which way to the information age?)," *IEEE Commun. Mag.*, vol. 24, no. 10, 1986.

[13] Bellcore, "Generic system requirements in support of switched multi-megabit data service," TA-TSY-000772, iss. 3, Oct. 1989.

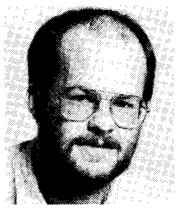
[14] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[15] V. B. Iversen, "Source policing in ATM-networks," in *Proc. Eighth Nordic Teletraffic Sem.*, Finland, 1989, session 2.2.

[16] K. Moth, "Broadband switching," Ph.D. dissertation, Electromagnetics Institute, Tech. Univ. of Denmark, Lyngby, Denmark, Dec. 1987.

[17] N. H. Sheng *et al.*, "A high speed 1 K-bit high electron mobility transistor static RAM," in *Proc. GaAs IC Symp.*, 1986, pp. 97-100.

[18] W. Whitt, "Approximation a point process by a renewal process: Two basic methods," *Operat. Res.*, vol. 30, no. 1, pp. 125-147, Jan.-Feb. 1982.

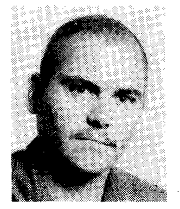


Lars Dittmann received the M.Sc.EE. degree in electrical engineering from the Electromagnetics Institute, Technical University of Denmark, Lyngby, Denmark, in 1987.

Since that time, he has been employed at the Centre for Broadband Telecommunications, Electromagnetics Institute, Technical University of Denmark. His main interest is within the area of ATM switching network performances, evaluation, and implementation aspects.

Søren B. Jacobsen received the M.Sc. degree in mathematics from Aarhus University, Denmark, in January 1986.

During 1986, he was employed at the Electromagnetics Institute, Technical University of Denmark, Lyngby, Denmark. In 1987, he was temporarily with the Mathematics Departments at UCLA and M.I.T. He returned to the Electromagnetics Institute, Technical University of Denmark, in January 1988. Since February 1990, he has been with ELLEMTEL Telecommunications System Laboratories, Lund, Sweden. His interests include ATM, switching network, and traffic performance evaluation of integrated switching networks.



Klaus Moth received the M.Sc.EE. degree in electrical engineering and the Ph.D. degree from the Electromagnetics Institute, Technical University of Denmark, Lyngby, Denmark.

He was the Technical Manager of the RACE "ATMOSPHERIC" subgroup dealing with load control and resource allocation in ATM-based broadband networks until the end of 1989. In parallel, he was the Manager of the RACE "ATMOSPHERIC" activities at the Electromagnetics Institute. At present, he is a Senior Systems Engineer at Alcatel Kirk Aerospace Systems Division, Ballerup, Denmark.