A very simple algorithm for flow control on high speed networks via la Palice queueings. Description and analysis

Philippe Jacquet, Paul Mühlethaler

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: (1) 39 63 55 11

Rapports de Recherche

N° 1371

Programme 1
Architectures parallèles, Bases de données,
Réseaux et Systèmes

A VERY SIMPLE ALGORITHM FOR FLOW CONTROL ON HIGH SPEED NETWORKS VIA LA PALICE QUEUEINGS

DESCRIPTION AND ANALYSIS

Philippe JACQUET
Paul MUHLETHALER

Janvier 1991

RR-1371

A VERY SIMPLE ALGORITHM FOR FLOW CONTROL ON HIGH SPEED NETWORKS VIA LA PALICE QUEUEINGS
description and analysis

Philippe JACQUET and Paul MÜHLETHALER
INRIA, Rocquencourt
78153, Le Chesnay Cedex, FRANCE
June 1, 1990

Abstract: *In future high speed networks, stations are divided into high speed logic devices and low speed logic devices. High speed logic devices show lower integration facilities than slower speed device (memory capacity). Thus lower speed mass memories must be attached to high speed logic devices. But the discrepancy between speeds may lead to critical bottlenecks in these very powerful systems; overflow of high speed devices of small capacity is possible. In this note we investigate a very simple algorithm for flow control that relies on some properties of La Palice queueings. After a description of the algorithm we give a sketch of evaluation based on La Palice queueings analysis.*

UN ALGORITHME TRES SIMPLE POUR LE CONTROLE DE FLUX DANS LES RESEAUX A HAUTE VITESSE, INSPIRE DES FILES D'ATTENTE DE LA PALICE

Résumé: *Dans les réseaux à hautes vitesses du futur, les stations sont partagées entre les composants à haute vitesse et les composants à basse vitesse. Les composants à haute vitesse montrent de moins bonnes facilités d'intégration que les composants à basse vitesse (notamment au niveau des capacités de mémoire). Donc des mémoires de masse lentes doivent être directement attachées aux dispositifs à haute vitesse. Mais la différence entre les vitesses peut se réveler un goulot d'étranglement crucial dans ces systèmes très performants ; la saturation du dispositif à haute vitesse est possible. Dans cette note nous étudions un protocole de contrôle de flux très simple qui s'appuie sur certaines propriétés des files d'attente dite de La Palice. Après une description détaillée de l'algorithme nous donnons une évaluation schématique basée sur l'analyse des files d'attente de La Palice.*

Abstract: *In future high speed networks, stations are divided into high speed logic devices and low speed logic devices. High speed logic device directly monitor data stream on the medium and must therefore work at the same speed as this stream. Unfortunately high speed devices show lower integration facilities than slower speed device (memory capacity). Thus lower speed logic devices must, as soon as possible, extract packets from high speed interfaces and store them in low speed mass memories. The coincidence of low speed on one side and low memory capacity on the other side may lead to critical bottlenecks in these very performant systems. Let fix the time unit to be the period with which low speed device can handle multi-packet messages. Let also suppose that high speed memories can store up to $n-1$ messages. Let λ be the average number of messages arriving at a given station ($\lambda < 1$). In absence of flow control algorithm a classical result from queueing theory ranges the probability of overflow of high speed buffers within $O(\lambda^n)$. In this note we investigate a very simple algorithm for flow control that ranges the probability of overflow within $O(\lambda^{n(n+1)(n+2)/6})$. The algorithm performances rely on some properties of so-called La Palice queueings. We say we have La Palice queueings when customers arrive in queue separated by more than their service time. We actually consider La Palice queueings with stochastic perturbations. After a description of the algorithm we give a sketch of evaluation based on La Palice queueings analysis. In appendix we investigate network architectures that fit the requirements of our algorithm.*

I. Introduction

High speed devices (AsGa chips and next generation of optical chips) present lower integration facilities in comparison with slower devices (classical Silicon chips), for prices that are usually for several orders of magnitude, higher. Therefore in optical networks, the high speed interface in station must be connected to low speed layers at very low level. But the gap between low speed and high speed is large and called to increase. This gap is one of most crucial challenge in the technology of modern communications.

Briefly the problem is "how fast can we extract packets from high speed buffers with low speed devices"? If you can do it at the same speed than packets arrive in the high speed buffers and if you think that things will remain like that in the future you can throw away this article. In fact it is reality in present architectures. But it seems that in the trends towards future architectures this balance won't be maintained. Of course there are plenty of propositions about how you can speed up your Silicon processor, how many you can put in parallel, and how cleverly you can schedule their tasks, that will little delay the moment where high speeds and low speeds will definitely clash. But something is clear in the story: Silicon processor performance painfully crawled from 10 MIPS to 100 MIPS (for some very specialized utilization, current applications are around a maximum speed of 20 MIPS) during the period of time where network speeds jumped over the gap between 10 Mbit/s to 1Gbit/s. Dreams about massive parallelism or pipe computations also call for soft and hardware limitations (you cannot put one processor and one bus for each bit in a packet, and several operations must remain sequential) [4].

Let us clearly state the problem. For convenience of presentation let us suppose that time is slotted on the optical medium. Let us call such slots, little slots. Packets fit little slots when they are transmitted on the optical medium. High speed device selects packets on the medium that are addressed to its host station and store them in high speed buffers

(or something related to). Let us now suppose that low speed device can extract packet from the high speed interface only every, say, ten little slots. Packets are usually parts of multi packet messages (of average say, ten packets). In a sender station, the message ready for transmission is stored in packets sequence in low speed memories. A low speed device will extract the packets from these memories and drop them into the high speed interface at a frequency that must be close to those of the reverse operation, say, one packet every ten little slots. Let us call the time needed for transmitting a multi packet message, a big slot (around hundred little slots in our model). In the following we will forget the delays due to packet contention on the optical medium. If only one sender sends a multi packet message to a given receiver at any given time, packets won't accumulate in the receiver high speed interface since they leave at the same speed they arrive. Problems may occur if a receiver has two simultaneous sources. Packets will arrive in the receiver high speed interface at a frequency twice the frequency they can leave, and, if messages are ten packet long, at the end of the big slot around ten packets will remain in the high speed buffer.

The question is what will happen if high speed buffers overflow? Incoming packets will be lost or at best deflected if the network technology allows it. Lost packets are usually expensive to recover since they call functions of higher level.

First, let us suppose that we don't care about this above since we believe having buffers large enough. Suppose that our high speed buffers can store up to n messages (if one packet 1Kbit, one message, around 10Kbit, and remember high speed memories are expensive and small). Let suppose that a given receiver receives messages from uncorrelated sources at a global rate of λ message per big slot without flow control. Message arrival in high speed buffer will match a Poisson distribution of parameter λ, and like in a M/D/1 queueing system the probability of overflow will be within $O(\lambda^n)$. If λ is around 1/10 or 1/3, and if we call to packet loss occurrence less than 10^{-9} [5] we shall need high speed buffers with a capacity of around twenty messages.

The first solution we may think about, is the classic flow control algorithm [11]. When a source has a multipacket to send, it first sends to the destination a request packet and the destination sends back a permission-to-transmit packet when it is ready to receive (enough room in the buffers). Unfortunately this solution is not good when we consider very high speed and wide-area networks: for each message the algorithm locks the receiver during a round trip delay that may seriously hurt its global reception ability.

In the next section we present our algorithm that is simply an extrapolation of the classic flow control algorithm. We will be shown in the third section that the overflow occurrence is lowered to $O(\lambda^{n(n+1)(n+2)/6})$ (therefore in the example above, four message capacity buffers may suffice). An intermediate algorithm will be presented that cancels overflow occurrence, but repeated requests occur with probability $O(\lambda^{n(n+1)/2})$ per multi packet message.

II. Specifications of the algorithm

Let us somehow precise the high speed interface architecture. It is mainly divided into output buffers, that store packets to be transmitted on the optical medium, and input buffers, that stores addressed packets selected from the optical medium. The algorithm is natural and simple. It relyes on the emission of a request packet from the sender and an answer packet from the receiver. But the protocol needs that the round trip delay between

any two pairs of station be the same. For example two topologies fit this condition: Cambridge Ring and Expressnet (see appendix, for details). In these very topologies the round trip delay that we define as the sum of the propagation delays for the request from the sender to the receiver and for answer from the receiver to the sender is equal to a constant which does not depend of the couple source destination.
Below are the rules of our algorithm, we append some comment after them.

1. A station which has a multi packet message to send, first drops into its output high speed buffer a request packet to be sent to the destination. This request packet contains the expected duration L that will be needed for the transmission of the whole message (the big slot duration). This duration is expressed in little slots unit.
2. Each destination monitors a load counter a which behaves like a load counter in a FIFO queue. At each little slot, if $a > 0$ then 1 is removed from a; if $a = 0$ then a remains at zero. When a request packet concerning a L multi packet message is received at the input high speed buffer, high speed device exports the current value of counter a back to the sender by creating a specific answer packet. The answer packet is directly stored in the high speed output buffer and then L is added to a.
3. To each of its pending messages the sender attaches a schedule counter b. When the sender receives an answer packet for a given message, it sets the value of b with the received value of a, and starts decrementing b every little slot. When $b = 0$, it sends the concerned multi packet message in the queue of messages to be dropped into the high speed output buffer.

There are little comment about rule 1. Of course, the request packet may contains additional informations about the multi-packet message. For example, the request packet may simply be the first packet of the message, if you want to save useful bandwidth. But, regarding our flow control algorithm, quantity L is the most important parameter to be contained in. This quantity can be simply computed by using the number of packets in the message and the frequency of the low speed device that is supposed to be known (unless constant on the network). It can also be a rough upper bound of this duration in order to overhelm transmission delays due to the contention on the medium.

It is also important that the packet request concerning the next multi packet message to be transmitted occurs at least L little slots after the transmission of this first request packet (theorem 0?). About rule 2, the latency between the time when a request packet is received and the time when the answer packet is transmitted, must be as small as possible, therefore the high speed interface must detect any request packet in its input buffer and catch quantity L involved, it has also to create itself the answer packet that should contain no other useful information than quantity a (with a few references about the multi packet message to be received). Therefore, about rule 3, the sender high speed interface must be able to detect any answer packet in its input buffer, and to catch quantity a (and the reference to the multi packet message) before destroying the packet. In other words, the answer packet is a way of communicating information only between two remote high speed interfaces. If a sender has several message which gave non zero a's, the high speed interface may also monitor more than one scheduled time at a time.

II.1 Local transmission conflicts on the source

There is a special comment about rule 3. The last sentence may be somewhat elliptic: "when $b = 0$, it sends the concerned multi-packet message in the *queue* of messages to be dropped in the high speed output buffer", and needs special explanations. The low speed device that drops packets into the high speed output buffer may be seen like a FIFO queueing system that serves message. When an answer packet has been received, and when the schedule counter, b, reaches zero, the high speed interface orders the queueing of the concerned message into this FIFO system. This message may not be alone on its host station and some other message may be also waiting for or currently being dropped into the output buffer (for other destinations, for example). In that case we say there is a local transmission conflict and the host station resolves it *via* FIFO policy. This kind of conflict may not be confused with eventual general packet collision on the optical medium which leads to some other problems. We will call this specific queueing system which stores message to be dropped in the high speed output buffer, the low speed output buffer.

A local transmission conflict is a bad event since it delays the transmission of a multi-packet message from the time which has been scheduled by its destination. The smaller is the transmission delay from that time, the better things will happen during its remote reception. This is why it is necessary to separate request packet by the corresponding big slot duration. Two consecutive messages with distinct destinations may, at low load, both receive a scheduled time equal to zero. Therefore if their request packet were sent at the same time, the messages will experience a local transmission conflict. This is not the case if their requests are separated by a big slot. Note that, in this model transmission conflicts are impossible if the destinations are the same (the remote destination will properly schedule the transmission times).

II.2 Local reception conflicts on the destination

If all packets are transmitted within their scheduled time, there will never be reception conflict on the destination interface: messages always arrive single. If the transmission delays experienced by the arriving messages due to local transmission conflict are too large, there will be possible reception conflicts. For some times, the frequency of packet arrival will overhelm the extraction frequency on the input high speed buffer, and an overflow remains possible. We will show in the next section that the probability of such event is *very* small with our algorithm.

III. Mathematical model

III.1 General assumptions

Our purpose is not to give an exhaustive and exact performance evaluation of the algorithm in specific cases of utilization. We only want to give some general insight of the behavior of the algorithm *via* a simple model.

We suppose that messages are all of the same length and stations are all synchronized within this uniform big slot. From now, big slot will be the time unit. We suppose that message generation on the sender station follows independent Bernoulli distribution of parameter λ on each big slot.

We assume that the network has a large number of connected stations so that the following is true: messages have uniform and independent destination in the sense that during any short enough time interval the message destinations scored on any given sender are all distinct and uncorrelated. Note that this kind of assumption may be seen as a *worst case* for our algorithm, since correlated destinations or favorite destination on short time interval may occur in real life and favor our algorithm with a better repartition of the scheduled times. We also suppose the same property for any given receiver regarding the sources of its addressed messages. The property is also true when regarding during a short period of time all the stations that are related to a given station by a finite chain of messages, given the length of such a chain be short enough.

For convenience of presentation we suppose that for any given receiver, the distribution of the number of messages generated during a given big slot and with this receiver as destination, is Poisson of parameter λ (symmetrical assumption).

III.2 Mathematical principles

Our algorithm takes advantage of the following principles.

Theorem 0 (*Vérité de Monsieur de La Palice*)
In any FIFO system, when the customers arrive separated by more than their service time, no queueing occurs.

This theorem can be found in Kermazov [1], but with a different wording it undoubtedly finds its origin in French history with Monsieur de La Palice (who was alive two hours before his death). Less obvious is the following theorem that we borrow from the famous joint book of Tcherniov, Havos and Way [2].

Theorem 1 (on the attracting property of the La Palice FIFO)
If we perturb a La Palice FIFO by delaying customer arrivals with independent delays which follow a distribution D_0, the inducted delay experienced by customers in queueing will follow distribution D_1, such that $D_1 < D_0$ in distribution. If we repeat the same experimentation but with new distribution D_1 instead of D_0, we obtain distribution D_2 and so forth we define the sequence of distributions D_n. We have the convergence in distribution:

$$\lim_{n \to \infty} D_n = 0 \, .$$

The proof of the above proposition was not constructive since it relied on very general assumption. We propose in the next section to give a quantitative proof that fits our scope of analysis.

Let us elaborate this very point. In our model, the low speed output buffer is a La Palice FIFO when all scheduled times are zero. Therefore, the distribution of the scheduled time plays the role of D_0 in theorem 1. The distribution of message delays due to local transmission conflict is therefore D_1. The high speed input buffer is also La Palice FIFO when the messages experienced no local transmission conflicts at the source. The delays due to local transmission conflict distribute like former D_1. At the very end of our analysis we are interested in the delays in the input high speed buffer which distribute like D_2.

III.3 estimating delays in local transmission conflict

We propose first to evaluate the scheduling time, namely distribution D_0. Let us consider a given receiver. According to our model sources of message are numerous and uncorrelated. If the big slot is large enough we can approximate the request packet arrival by a Poisson process of rate λ per big slot. The counter a is nothing else than the size of a virtual M/D/1 queueing system. We suppose that delays are integer slots. The following theorem holds.

Theorem 2
The distribution D_0 has the following generating function

$$d_0(z) = \frac{(1-\lambda)}{\lambda} \frac{(\exp(\lambda(z-1)) - 1)}{(z - \exp(\lambda(z-1)))} .$$

Proof: This is the classical generating function of delays for a M/D/1 queueing. ∎

Note, as a classical result of queueing analysis, that the probability that the value of the scheduled time be i big slots is $O(\lambda^i)$.

Now we are attacking the corner stone of our analysis, the evaluation of distribution D_1 from distribution D_0. In fact the further evaluation of distribution D_2 will be a by product of this first analysis.

In the following we suppose that the physical propagation delay and computation time are zero (unless it is a constant time shift), and we suppose that the reception of the answer packet occurs at the same time the request packet is sent. Let ξ be a row vector of the form $[\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_i, \ldots]$ where all the ε_i's are 0 or 1, and where the 1 are in a finite number. Let n be an integer, and let $q(n, \xi)$ be the stationary probability of a very special event. This event is the coincidence of the following events:

1. the queue in the low speed buffer is exactly of length n;
2. i big slots in the past there was ε_i message which has been generated on the host station and such that the answer packet gave a scheduled time greater than or equal to i ($a \geq i$).

Note that the latter is consistent with constraint $\varepsilon_i = 0$ or 1, since message generation is assumed Bernoulli per big slot. Note also that the process is Markov within the states (n, ξ). The difficulty of the approach is in the fact that we must keep some track of the past events with the vector ξ. Let χ be an infinite real or complex row vector of the form $[x_1, x_2, \ldots, x_i, \ldots]$, and z be a real or complex number, we define the generating function $Q(z, \chi)$ by

$$Q(z, \chi) = \sum_{n,\xi} q(n, \xi) z^n \chi^\xi ,$$

with the convention $\chi^\xi = x_1^{\varepsilon_1} x_2^{\varepsilon_2} \cdots x_i^{\varepsilon_i} \cdots$. For $i \geq 1$, we note p_i the probability that a random scheduled time be strictly greater than $i-1$ given that it is greater than or equal to $i-1$. Note the p_i's are $O(\lambda)$. The quantity $p_1 p_2 \cdots p_{i-1}(1-p_i)$ is the probability that the scheduled time be unconditionally equal to $i-1$.

We will note by 1 the specific vector $[1, 1, \ldots]$ when no confusion will be possible. We note by σ the shift operator on vectors $\sigma(\chi) = [x_2, x_3, \ldots]$. We also note π the specific

operator such that $\pi(\chi) = [p_1 x_1, \ldots, p_i x_i, \ldots]$. On generating function $f(z,\chi)$ we note S the operator defined by

$$Sf(z,\chi) = \frac{f(z,\chi) - f(0,\chi)}{z} + f(0,\chi) \ .$$

Theorem 3
The generating function $Q(z,\chi)$ satisfies the following functional equation:

$$Q(z,\chi) = SQ(z, z(1-\pi(1)) + \pi(\sigma(\chi))) \times (1 - \lambda + \lambda x_1) \ .$$

Proof: Note that $S(z^n) = z^{n-1}$ if $n \geq 1$ and $S(z^0) = z^0$, therefore $SQ(z,\chi)$ is the generating function of our Markov process just after service in the buffer, since $Q(z,\chi)$ is the generating function just *before*. The extended expression for vector $z(1-\pi(1))+\pi(\sigma(\chi))$ is exactly $[(1-p_1)z + p_1 x_2, \ldots, (1-p_i)z + p_i x_{i+1}, \ldots]$ which is the translation in terms of generating function of the fact that the messages, which are currently decrementing their scheduled time, independently experience the following alternative. We consider a message which has been generated i big slots in the past; either it waits for one big slot more (with probability p_i), either it immediately enters the buffer because of the expiration of its scheduled time (with probability $1-p_i$). The multiplication by $(1-\lambda+\lambda x_1)$ denotes eventual arrival on the current big slot (following a Bernoulli process). ∎

For convenience of presentation we will denote by P the operator on generating function defined by

$$Pf(z,\chi) = Sf(z, z(1-\pi(1)) + \pi(\sigma(\chi))) \times (1 - \lambda + \lambda x_1) \ .$$

Generating function $d_0(z)$ is analytic with respect to variable λ and therefore p_i are also analytic in λ. One may expect the same from generating function $Q(z,\chi)$ which thus is a linear combination of monomials of the form $z^n \chi^\xi \lambda^\ell$. For the following we note by $|\xi|$ the number of 1 in ξ, in other words $|\xi| = \sum_{i=1}^{\infty} \varepsilon_i$. For n, integer, we note by $|n|_2$ the quantity $n(n+1)/2$; note that $|n|_2 = |n-1|_2 + n$. Let ζ be the vector $[\nu_1, \nu_2, \ldots]$, we use the notation $\zeta \leq \xi$ when for all i: $\nu_i \leq \varepsilon_i$; in this very case we can make use of the notation $\xi - \zeta$ for the vector $[\varepsilon_1 - \nu_1, \ldots, \varepsilon_i - \nu_i, \ldots]$.

Theorem 4
The generating function $Q(z,\chi)$ is a linear combination of monomials $z^n \chi^\xi \lambda^\ell$ such that $\ell \geq |n + |\sigma(\xi)||_2 + |\xi|$.
Proof: Let us call such monomials, two-order monomials. In order to prove the theorem it suffices to establish that $P(z^n \chi^\xi)\lambda^{|n+|\sigma(\xi)||_2+|\xi|}$ is a linear combination of two-order monomials. First let us prove the proposition for $n > 0$. We have

$$P(z^n \chi^\xi) = \sum_{\sigma(\zeta) \leq \xi} z^{n-1} [(1-\pi(1))z]^{\xi - \sigma(\zeta)} [\pi(\sigma(\chi))]^{\sigma(\zeta)} [x_1 \lambda]^{\nu_1} (1-\lambda)^{1-\nu_1} \ .$$

Let us consider one of this monomial in ζ (such that $\sigma(\zeta) \leq \xi$) and let us multiply it by $\lambda^{|n+|\sigma(\xi)||_2+|\xi|}$. The result can be divided by the simpler

$$z^{n-1+|\xi-\sigma(\zeta)|} \chi^\zeta \lambda^{|\zeta|+|n+|\sigma(\xi)||_2+|\xi|} \ ,$$

since $\pi(1) = O(\lambda)$, *i.e* each p_i is analytic with respect to λ without coefficient of degree 0. We will prove that the monomial is two-order if we show that

$$|n-1+|\xi-\sigma(\zeta)|+|\sigma(\zeta)||_2+|\zeta| \leq |\zeta|+|n+|\sigma(\xi)||_2+|\xi|\ ,$$

which is easy to establish since $|\xi-\sigma(\zeta)| = |\xi|-|\sigma(\zeta)|$ and $|\sigma(\xi)| \geq |\xi|-1$.
If we consider $n = 0$, the problem now is about the monomial

$$z^{|\xi-\sigma(\zeta)|}\chi^{\zeta}\lambda^{|\zeta|+||\sigma(\xi)||_2+|\xi|}\ ,$$

and we must show that $||\xi-\sigma(\zeta)|+|\sigma(\zeta)||_2+|\zeta| \leq |\zeta|+||\sigma(\xi)||_2+|\xi|$. This last inequality is a little more delicate to deal with than those when $n > 0$, but we know that $|\sigma(\xi)| \geq |\xi|-1$ and $||\xi|-1|_2+|\xi| = ||\xi||_2$. ∎

Corollary 5
The unconditional probability that the low speed buffer queues n messages at the same time is $O(\lambda^{n(n+1)/2})$.
Proof: In fact theorem 4 directly shows the proposition uniformly for $\lambda < 1$ but not uniformly for n. For this last point a more involved analysis should be necessary that may exceed the span of this paper. Note that the estimate $O(\lambda^{|n|_2})$ is optimal since the event of n consecutive messages with respectively $0, 1, \ldots, n-1$ as scheduled times leads to queue size, n, and occurs with probability $p_1^{n-1}p_2^{n-2}\cdots p_{n-1}\cdot(1-p_1)(1-p_2)\cdots(1-p_n)\lambda^n$. ∎

Theorem 6
The unconditional probability that a message experiences n big slots in local transmission conflict is $O(\lambda^{n(n+1)/2-1})$, and the generating function of this distribution, D_1, is

$$d_1(z) = \frac{Q(z,1)-SQ(z,1)}{\lambda(z-1)}\ .$$

Proof: The first statement is a direct corollary of corollary 5, since the probability that the queue length be greater than n is an upper bound of the probability of experiencing delay n for a random message. Note that in this delay we include the own transmission time of the message. If we don't want to include it the estimate becomes $O(\lambda^{n(n+3)/2})$. The second statement is more technical.

Let us consider state (n, ξ) at big slot number t. Our aim is to derive the generating function of the delays that the messages will experience which enter the buffer at big slot $t+1$. Between slot t and $t+1$ the buffer service makes the number of messages in buffer decrementing to $n-1$, except the special case where $n = 0$. Let m be the number of message which enter the buffer during the new coming slot. We have $m \leq |\xi|$ and this number is linked to vector ξ by its generating function, $[(1-\pi(1))z+\pi(1)]^{\xi}$. The m messages will wait respectively for $n-1, n, \ldots, n+m-1$ big slots in buffer, leading to a generating function $z^{n-1}(1+z+\cdots+z^{m-1}) = z^{n-1}(z^m-1)/(z-1)$ (more precisely $S(z^n)(z^m-1)/(z-1)$. Therefore state (n, ξ) leads to the delay generating function $(S(z^n)[(1-\pi(1))z+\pi(1)]^{\xi}-S(z^n))(z-1)^{-1}$. Summing over all possible state (n, ξ) we get $(PQ(z,1)-SQ(z,1))(z-1)^{-1}$ as the generating function of the delays experienced by

messages that enter buffer at a given random big slot. With $PQ(z) = Q(z)$ and dividing by λ, since the mean number of new generated message is exactly λ per big slot, the theorem is proved. ∎

III.4 Estimating queue size in reception conflicts

In the following we will estimate the queue size of the input buffer of the receiver, given that arriving messages are delayed according to distribution D_1. If messages were not delayed, they should arrive in the buffer separated by their service time therefore the buffer is a La Palice FIFO. One detail is the fact that message should arrive according to their scheduled time, that is scheduled as if they were issued from an upper FIFO. This kind of arrival process differs from a Bernoulli process. For simplicity of the presentation we will suppose that message are scheduled like a Bernoulli process, in order to fit the previous sub-section. This kind of assumption won't change the magnitude order of our estimates.

Therefore to estimate the input buffer queue size we will consider the same problem as in previous section, but with the new perturbation delay distribution, D_1, instead of D_0.

We keep the same notation as in the previous section. Note that now the p_i's are $O(\lambda^i)$. And we want to estimate new generating function $Q(z, \chi)$. Theorem 3 holds in his previous form. There is a new theorem 4. Let $|n|_3 = n(n+1)(n+2)/6$, note that $|n|_3 = |n-1|_3 + |n|_2$; for any vector ξ let $|\xi|_2$ be an abusive notation of $\sum_{i=1}^{\infty} i\varepsilon_i$, we will use it each time no confusion with notation $|n|_2$, with n integer, is possible. Note that $|\xi|_2 \geq ||\xi||_2$.

theorem 4 bis

The generating function $Q(z, \chi)$ is a linear combination of monomials $z^n \chi^{\xi} \lambda^{\ell}$ such that $\ell \geq |n + |\sigma(\xi)||_3 + |\xi|_2$.

Proof: Let us call such monomials, three-order monomials. Following the same proof as for theorem 4, we take advantage of the fact that $\lambda^{\nu_1} \pi(1)^{\sigma(\zeta)}$ can be divided by $\lambda^{|\zeta|_2}$, since the p_i's are $O(\lambda^{i+1})$. Therefore we have to prove that $|n - 1 + |\xi - \sigma(\zeta)| + |\sigma(\zeta)||_3 + |\zeta|_2 \leq |\zeta|_2 + |n + |\sigma(\xi)||_3 + |\xi|_2$, when $n > 0$, which is straightforward. When $n = 0$ the inequality becomes $||\xi - \sigma(\zeta)| + |\sigma(\zeta)||_3 + |\zeta|_2 \leq |\zeta|_2 + ||\sigma(\xi)||_3 + |\xi|_2$ which is done noticing $||\sigma(\xi)||_3 + |\xi|_2 \geq ||\xi| - 1|_3 + ||\xi||_2 = ||\xi||_3$. ∎

Corollary 5 bis

The unconditional probability that the input high speed buffer queues n messages at the same time is $O(\lambda^{n(n+1)(n+3)/6})$.

Note that this bound is optimal for similar reasons as those developed in the previous section.

Corollary 7

The probability that a high speed input buffer of capacity $n-1$ overflows is $O(\lambda^{n(n+1)(n+3)/6})$.

Just for lazy readers who don't want to open the left hand slider of their desk and catch

their pocket calculator, we show below, quantities $|n|_2$ and $|n|_3$ for some value of n.

n	$\|n\|_2$	$\|n\|_3$
1	1	1
2	3	4
3	6	10
4	10	20
5	15	35
6	21	56
7	28	84
8	36	120
9	45	165
10	55	220
15	120	680
20	210	1540

III.5 Computing generating functions

Computing generating function $Q(z,\chi)$ is an important problem. Since χ is an infinite dimension vector, direct computing leads to inextricable combinatorial problems. The way we choose to cope with this difficulty consists in taking advantage of respectively theorems 4 and 4 *bis*: generating function $Q(z,\chi)$ is a linear combination of two-order (respectively three-order) monomials in $z^n\chi^\xi\lambda^\ell$. Therefore, if we limit the Taylor expansion of Q with respect to variable λ, we will at the same time limit the number of monomials in a way that allows reasonably short computations to an accuracy around λ^{30}. The scheme is based on iteration $Q_{n+1} = PQ_n$ starting from $Q_0(z,\chi) = 1$. A classic result in asymptotic analysis tells that the number $p(k)$, of vectors ξ such that $|\xi|_2 = \sum_{i=1}^{\infty} i\varepsilon_i \leq k$, for a given k is asymptotically equivalent to

$$p(k) \sim \frac{1}{4\sqrt{2}\cdot 3^{1/4}k^{3/4}} \exp(\pi\sqrt{k/3} + \frac{1}{2} - \frac{\pi^2}{12}) .$$

For $k = 30$, $p(30) \sim 300$. The number $p(k)$ is related to the well known partitions number of k, and the asymptotic expansion techniques are due to Hardy and Ramanujan (improved by Rademacher [3]). Results of the computations are shown in figures 1 and 2 which show high speed buffer overflow probability *versus* λ for different buffer capacities with or without flow control algorithm.

IV An intermediate algorithm

Enlightened by the analysis showed in the previous section, we can easily build an algorithm that should cancel overflow occurrence. Let us just suppose that low speed frequency and high speed buffer capacity be uniform among all connected station on the network (that however should be normalized feature). Let $L_{\max}$ be translation in transmission duration of this capacity; if W is the frequency of packet extraction per time unit, if n is the capacity of high speed buffer, in number of packet, we have $L_{\max} = n \cdot W^{-1}$. The intermediate algorithm simply consists in not transmitting messages that experienced delay in transmission

conflict greater than $L_{\max} - L$, where L denotes the expected transmission duration of the message itself. Thus, messages that overtook this deadlines are just dequeued from low speed output buffer (in which messages are simply denoted by a token indicating a low speed memory field), and must be addressed for a new request.

IV.1 Analytic proofs about the intermediate algorithm

In this subsection we give some sketch of the proof about the fact that the algorithm *really* cancels overflow occurrence. Note that the proof may directly be derived from theorem 1 with some minor restriction: D_0 must include service time and so do D_1. But our purpose is to give a self contained proof of this phenomenon. For simplicity of presentation let us adopt one more time the mathematical modelization of section III.

Let us suppose that k is the capacity of high speed input buffers. Therefore delays in transmission conflict are limited up to $k-1$ big slots. In other words we have $p_k = 0$. Therefore vector ξ cannot have 1 after the kth position, for all $i > k$, $\varepsilon_i = 0$.
Let us now show a simple proof.

Let, at a given time, be (n, ξ) the state of our system. It is obvious that $n + |\xi|$ increases only if $n = 0$ and an arrival occurs, in that case the increment is of one unit. If no arrival occurs and $n > 0$, quantity $n + |\xi|$ decreases of one unity. In the other cases $n + |\xi|$ remains steady.

Let us consider a period separating two consecutive incursions in states of the form $(0, \xi)$. Whatever be the state we know that $|\xi| \leq k$. Let (m, ζ) denote the next states attained during that period. We have $m > 0$ by hypothesis, therefore quantity $m + |\zeta|$ cannot increase inside the period. Since it can increase only at the beginning of the period, we have thus $m + |\zeta| \leq k + 1$. On the other hand $m + |\zeta|$ will decrease each time $\zeta = 0$. Therefore we always have $m \leq k$. ∎
Note that a proof in continuous time or in general message length is possible, but we skip it for keeping the paper in a reasonable length. However, the following holds.

Theorem 8
The probability that a message experiences more than one request with the intermediate algorithm when the high speed input buffer is of capacity $n-1$ is $O(\lambda^{n(n+1)/2-1})$.
Proof: Taking the results from section III, it is exactly the probability of experiencing $n-1$ big slots or more in low speed output buffer. ∎

V. A TEMPTING SOLUTION

In this section we investigate a tempting feature that we can append to our algorithm. The feature is one of the most popular propositions you may find in literature about high speed networking and it occurs in so many papers that one should offer some medal (in Patagonium, for example) to those who don't mention it. Since this medal does not exist yet, there is no reason why we shall not follow the slope of facility.

V.1 Slow down the packet emission frequency per message

Suppose that the frequency that allows low speed packet extraction from high speed buffer is one packet per 10 little slots. Suppose now that we slow down the frequency emission of each message to be at one packet per 20 little slots. In that case the following holds:

(i) the low speed output buffer can simultaneously drop two messages in the high speed interface;
(ii) two messages can be simultaneously extracted from the high speed input buffer, in the sense that the packet extraction frequency is twice the packet arrival frequency per message.

Let us suppose that we maintain the same algorithm, the big slots are now twice bigger than in the previous section, and the La Palice FIFO we will consider are related to queueing systems with double servers. In counterpart the message arrival rate per big slot increases to 2λ. In the following we consider the general case where we slow down packet emission frequency per message by an integer factor k. We suppose that $k \geq 1$. We will show that the algorithm notably improves its performance when $k \geq 2$, given that $\lambda < \frac{1}{k}$.

V.2 Analytical model

We take the same modelization as in section III. Note that now big slots are physically k times longer than before. We suppose that $\lambda < \frac{1}{k}$.

Theorem 9
The distribution D_0 has the following generating function

$$d_0(z) = \frac{(1-k\lambda)}{k\lambda} \frac{(\exp(k\lambda(z-1))-1)}{(z-\exp(k\lambda(z-1)))} .$$

Proof: message arrival rate per big slot is now $k\lambda$. ∎

Let us introduce the operator S_k defined on generating function $f(z)$. Let us suppose that $f(z) = \sum_{i=0}^{\infty} a_i z^i$, we define

$$S_k f(z) = \frac{f(z) - \sum_{i=0}^{i=k-1} a_i z^i}{z^k} + \sum_{i=0}^{i=k-1} a_i .$$

Note that $S_1 = S$ as defined in section III.

Theorem 10
The generating function $Q(z,\chi)$ related to the low speed output buffer satisfies the following functional equation

$$Q(z,\chi) = S_k Q(z, z(1-\pi(1)) + \pi(\sigma(\chi))) \times (1 - \lambda + \lambda x_1) ,$$

where the p_i's are $O(k\lambda)$.
Proof: The operator S_k is the service operator related to k-server queueings. The p_i's are computed via theorem 9 and are $O(k\lambda)$ *via* a classical result in queueing theory. ∎

Theorem 11
The unconditional probability that the low speed buffer queues n messages at the same time is $O([k\lambda]^{n(n+1)/2})$.

Proof: This estimate is an obvious upper bound since the homothetic transformation of λ into $k\lambda$ in the basic algorithm gives an upper bound to the new algorithm. But this estimate is optimal since the occurrence of n consecutive messages with respective scheduled times $n-1, n-2, \ldots, 0$ occurs now with probability $p_1^{n-1}p_2^{n-2}\cdots p_{n-1}\cdot(1-p_1)(1-p_2)\cdots(1-p_n)\lambda^n$, which is $O([k\lambda]^{|n|_2})$. ∎

Corollary 12
The unconditional probability that a message experience n big slots in local transmission conflict is of the order of $k\lambda$ raised to the power of $(nk+1-k)(nk+2-k)/2-1$.
Proof: In order to experience n big slots in low speed output buffer (including the own transmission time), the message needs to belong to a queue of minimal length $(n-1)k+1$. Excluding the own transmission time the probability becomes $O([k\lambda]^{kn(kn+3)/2})$. ∎

Theorem 13
The generating function $Q(z,\chi)$ related to the high speed input buffer satisfies the following functional equation

$$Q(z,\chi) = S_k Q(z, z(1-\pi(1)) + \pi(\sigma(\chi))) \times (1-\lambda+\lambda x_1)\ ,$$

where the p_i's are $O([k\lambda]^{(ik+(3-k)/2))k})$.
Proof: We assume approximation adopted in subsection III.4 that suppose messages scheduled according to a Bernoulli distribution. The essential novelty of this theorem comes in fact from corollary 12 which leads to the estimate $p_i = O([k\lambda]^{(ik+(3-k)/2)k})$. ∎

Theorem 14
The probability that the high speed input buffer holds n messages at the same time is of the order of $k\lambda$ raised to the power of $k^2|n|_3 - |n|_2 3(k^2-k)/2 + n(k-1)(k-2)/2$.
Proof: Classic event when n consecutive messages arrive with respective scheduled times $n-1, n-2, \ldots, 0$ occurs now with probability $p_1^{n-1}p_2^{n-2}\cdots p_{n-1}\cdot(1-p_1)(1-p_2)\cdots(1-p_n)\lambda^n$ which is of the order of magnitude of the estimate asserted in the theorem. ∎

Below we show estimate $k^2|n|_3 - |n|_2 3k(k-1)/2 + n(k-1)(k-2)/2$ for some values of n and k.

$n \setminus k$	1	2	3	4
1	1	1	1	1
2	4	7	11	16
3	10	22	39	61
4	20	50	94	152
5	35	95	185	305
6	56	161	321	536
7	84	252	511	861
8	120	372	764	1296
9	165	525	1089	1857
10	220	715	1495	2560
15	680	2360	5055	8765
20	1540	5530	11990	20920

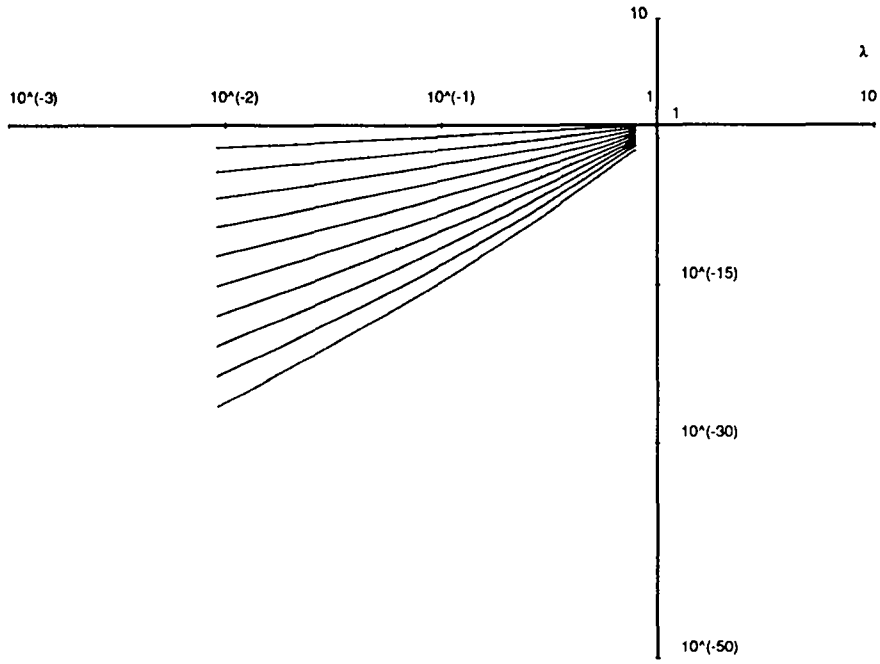

Figure 1: overflow probabilities *versus* λ without flow control
for different buffer sizes ranging from 1 to 10 message capacity

The figures above must be temporized by the fact that it is $k\lambda$ which is brought to such power and not λ alone. The table below shows for n and k given the critical value of λ below which it is worthy to increase packet emission period by the factor $k+1$ instead of k, when high speed buffer capacity is up to $n-1$ messages.

$n \setminus k$	1	2	3	4
2	0.19843	0.16395	0.13276	0.11031
3	0.28062	0.19724	0.15013	0.12081
4	0.31498	0.21027	0.15684	0.12487
5	0.33371	0.21727	0.16044	0.12705
6	0.34548	0.22166	0.16271	0.12842
7	0.35355	0.22467	0.16426	0.12937
8	0.35944	0.22687	0.16539	0.13006
9	0.36391	0.22854	0.16626	0.13058
10	0.36743	0.22986	0.16694	0.13100
15	0.37768	0.23371	0.16893	0.13221
20	0.38263	0.23558	0.16990	0.13280

VI. A GENERAL REMARK AND CONCLUSION

We stress again that our analysis actually gives an upper bound for overflow probability since our hypotheses can be considered as worst cases. However our algorithm already

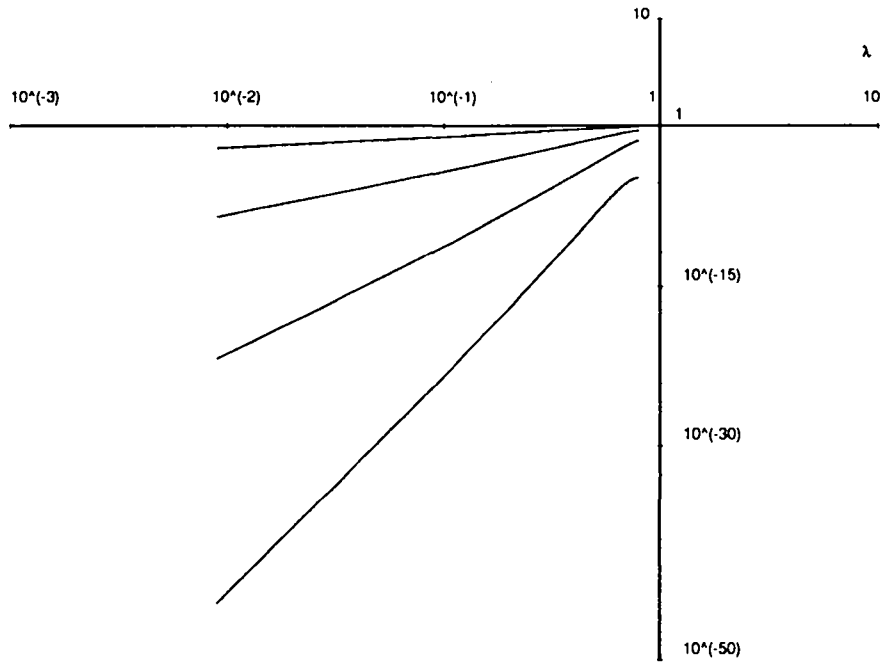

Figure 2: overflow probabilities *versus* λ with flow control
for different buffer sizes ranging from 1 to 4 message capacity

shows very promising performances. The first worst case we set in hypothesis is the assumption that consecutive messages on a given source have uncorrelated destinations that we know to be false since favorite destinations may be reported. The second worst case assumption in our analysis, is the fact that if n messages simultaneously arrive at a given destination, the high speed input must hold all of them at the end of the big slot. This fact forgets that messages arrive in small pieces (packet) and some of them are continuously extracted from the high speed input buffer. Optimistically one may assume that the high speed buffer holds $n-1$ instead of n messages at the end of the big slot. Therefore one may vertically shift one position in the tables shown in section III: the probability that a high speed buffer of capacity $n-2$ overflows is $O(\lambda^{|n|_3})$. In case where packet emission speed is slow down by a factor k, the buffer may holds $n-k$ messages at the end of a simultaneous reception of n messages. Therefore one may read tables in section V by shifting k vertical positions (diagonal reading): the probability that a high speed buffer holds $n-k-1$ messages is of the order of $k\lambda$ raised to the power of $k^2|n|_3 - |n|_2 3(k^2-k)/2 + n(k-1)(k-2)/2$.

In summary the algorithm we propose is simple and shows really good performance. The implementation of the algorithm technically allows to decorrelate low speed and high speed devices. This opportunity may free the latter to reach very high speeds in wide area networks of tomorrow.

Reference

[1] KERMAZOV, *Discours préliminaire et introductif à toute chose*, Vol. 16, Académie de

Saint Petersbourg, 1878.

[2] TCHERNIOV, HAVOS, WAY, "The incredible world of La Palice stochastic processes," in IEEI *Transaction on no noitcasnart*, number 2, pp 220, 230, 1956.

[3] G. ANDREWS, *The Theory of Partitions*, Encyclopedia of mathematics, 1976.

[4] H. MELEIS, A.TANTAWI, "High performance networking and the modular communication machine (MCM) approach," in the proceedings of Future Trends'90, pp. 298-304, 1990.

[5] M. EL ZARKI, S. BAHLK, "A distributed routing scheme for network of the future," –, pp. 81-87, 1990.

[6] R. FALCONER, J. ADAMS, "Orwell: a protocol for an integrated services local network," British Telecom. Technology Jour., Vol 3., pp. 27-35, 1985.

[7] FDDI Token Ring Media Access Control, draft proposed ANSI, X3T9/84-100, rev-9, 1985.

[8] DQDB, Distributed Queue Dual Bus, proposed standard for metropolitan area network, draft proposal IEEE 802.6/D6, 1988, rev. 1989, 1990.

[9] P. JACQUET, P. MÜHLETHALER, "Fairnet: a fair access protocol for very high speed networks," To appear.

[10] F. TOBAGI, F. BORGONOVO, L. FRATTA, "Expressnet: a high performance integrated-services local area network," IEEE Journal on selected areas in communications, vol sac-1, No 5, 1983.

APPENDIX

This section is devoted to the presentation of a few high speed network architectures that fit or do not fit the requirements of our simple flow control algorithm. We remind that the most important requirement that this architecture has to satisfy is the delay between generation of request-packet and the reception of the answer packet which has to be within the same for every pair source-destination. As "within the same" we mean that things have to be similar when regarding to an averaged big slot (say the discrepancy ranges within 10 little slots). This inaccuracy allows some random noise in delays.

Regarding delays between consecutive packet transmissions by the same sender it is also necessary that it fits the speed at which packet are dropped into the high speed output buffer.

A. FDDI and DQDB

FDDI [7] and DQDB [8] have topologies that do not fit requirements of our algorithm. DQDB first does not provide the same request-answer delay for any source-destination pair. DQDB, as Distributed Queue Dual Bus, relies on two busses in opposite stream; if a packet is sent on one bus the answer packet will be send on the opposite bus. If we omit contention delays and local computation latencies (regarded as random noise, or constant time shift) request-answer delay is proportional to the distance between source and destination.

FDDI is basically a token ring adapted to moderate high speed technology. As in any token based architecture, two consecutive packet transmissions by the same sender, are necessarily separated by a token rotation. Therefore, if the network is too wide or the number of stations is too important, this incompressible delay may exceed the low speed device period. In fact FFDI is dedicated to moderate speed networks (100 Mbit/s) and therefore is not concerned with the problem about tradeoff between high speed logic and low speed logic devices.

B. The Cambridge Ring and Fairnet

The Cambridge Fast Ring Networking System [6] is a slotted ring where many slots permanently rotate at the same speed. Stations contend for slots and fill them with packets. Packets fit the slot length and a specific bit indicates if the slot is empty or busy. The protocol simply allows for any station with a pending packet to use the next incoming free slot. The only rule consists in the fact that a sender have to itself remove its packet from the slot after complete rotation and reset the specific bit (also consider receiver release that save some bandwidth like in Orwell [6]). For fairness consideration, the sender is not allowed to re-use the same slot just after packet removal.

If we omit contention delays on medium and local computation latencies, the delay between request transmission and answer reception is always a round trip delay whatever is the source-destination pair. Contention delays regarded as random noise let the delay between two consecutive packet transmission by the same sender fit the low speed device period. Therefore the Cambridge Ring fits our algorithm.

Fairnet is a proposition of INRIA [9] which is a kind of adaptation of Cambridge Ring to very high speed architecture where packet removal is not easy to do. The topology is

like a "S" in Expressnet [10], the unidirectional medium is folded twice so that each station is connected twice: one tap on the outbound link and another tap on the inbound link. Packets are written on the outbound link and read on the inbound link. The "S" topology implies that the request-answer delay is always equal to two end-to-end propagation delays, if we omit contention delays.

The access protocol is similar to Cambridge Ring regarding basic mechanisms. Each slot carries two specific bits: an occupation bit and a release bit. The occupation bit indicates if the slot is free or busy. The release bit is more delicate to deal with. If a station reads one of its own packet on its inbound tap, it immediately sets to one the release bit of the next incoming slot at its outbound tap. Therefore a station with a pending packet waits for the next incoming free slot on the outbound link that fits one of the following alternatives. Either the slot read just before on the inbound link is empty, either it is busy but the release bit on the outbound link is already set to one. In fact Fairnet is a kind of Cambridge Ring where packet removal is finally done at the dead end of the inbound link.

Fairnet of course fits our flow control algorithm.

Imprimé en France
par
.l'Institut National de Recherche en Informatique et en Automatique.

ISSN 0249 - 6399