Real-Time Communication in Packet-Switching Wide-Area Networks

Domenico Ferrari

Computer Science Division
EECS Department
University of California
and International Computer Science Institute
Berkeley, California

Abstract

The increasing importance of distributed multimedia applications and the emergence of user interfaces based on digital audio and digital video will soon require that computer communication networks offer real-time services. This paper argues that the feasibility of providing performance guarantees in a packet-switching wide-area network should be investigated, and describes a possible approach. We present a model of the network to be studied, formulate the problem, give a definition of the guarantees to be provided, and describe a scheme for the establishment of real-time connections with deterministic, statistical, and best-effort delay bounds.

This research has been supported in part by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by Naval Electronic Systems Command under Contract No. N00039-84-C-0089, the University of California with a MICRO Program grant, Cray Research, Inc., Hitachi, Ltd., IBM Corporation, Ing. C. Olivetti & C., S.p.A., and the International Computer Science Institute. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring organizations or of the U.S. Government.

1. Introduction

The current evolutionary trends of computing and networking technologies point to a major increase, over the next several years, in the need for real-time communication, i.e., for computer communication with guaranteed performance. Performance guarantees are required not only in distributed process-control and military systems, but also in supercomputer-to-supercomputer communications and even in such business applications as stock trading. Furthermore, guarantees will be needed by all those systems that process and transmit digital audio, digital video, or any combinations of each with other types of information. Multimedia distributed systems are expected to emerge in the near future, and to become predominant shortly thereafter, as the newer audio- and video-based user interfaces are added to the traditional ones (data, text, and images).

Providing performance guarantees typically means guaranteeing that, given certain bounds on the flow of information into the network, the delay with which each information item reaches the destination will satisfy given bounds. For interactive audio and video, the bound will usually be on the value of the delay of each bit, or byte, or message transmitted by a given sender to a given receiver; for instance, it is well-known that, in a phone conversation, delays longer than half a second are sufficient to make the speakers uncomfortable. In the case of non-interactive audio and video communication (e.g., the transmission of music or a movie for immediate or deferred consumption by the receiving user), performance indices more important than the actual delay are its variance and, even more, its jitter. The delay jitter with which a packet arrives at its destination may be defined as the absolute value of the difference between the packet's actual delay and its target (i.e., desired, ideal) delay. Once a network or internetwork starts offering real-time services, other applications of these services are likely to emerge; for example, it may be desirable to have some remote procedure calls and their replies, and some accesses to remote files or databases, performed within certain maximum time intervals. The availability of real-time services, if their prices are not prohibitive, is likely to expand the demand for them well beyond the boundaries of those applications that are motivating their introduction.

Some of the solutions proposed for the problem of designing network protocols and management policies to support real-time communication apply to local-area networks. They include such media access protocols as the Fiber Distributed Data Interface (FDDI) and the High Speed Ring Bus (HSRB); both of them provide a guaranteed delay bound, FDDI using the timed token scheme [Ross86], and HSRB the reservation/priority mechanism [SAE86][1]. Circuit switching can be used for this purpose in wide-area networks as well as in LANs: by dedicating a circuit of fixed bandwidth to each connection for the entire duration of the connection we can bound delays. However, circuit switching is not the most convenient technique for data communication [Harr80]. Hence, schemes that combine packet and circuit switching features in various ways have been proposed for integrated voice/data networks: fast circuit switching, burst switching, hybrid switching, and fast packet switching. An overview of these schemes can be found in [Chen88]. The most promising of them is fast packet switching, which is based on a major simplification of the lower layers in the protocol hierarchy; this simplification yields real integration of services (at the lower layers, all packets are treated in the same way, no matter whether they contain data or voice, or perhaps even video) as well as the possibility of protocol implementation in hardware. It is important to notice, however, that fast packet switching can provide low but not necessarily bounded delays, and therefore does not per se offer true real-time communication.

[1] Many other approaches to providing some form of real-time communication can be described as *best-effort* methods, that is, methods in which the system tries to make sure that most messages will satisfy their deadlines. These approaches include virtual-time [Zhao87] and window-based [Kuro88] [Zhao88] access methods for CSMA/CD networks. [Stro89] provides delay guarantees by controlling access to a token ring network on the basis of prior knowledge about the arrival patterns of messages.

This paper addresses the problem of providing performance guarantees in a wide-area network or internetwork. The solution we present applies to all current and future networks that are adequately represented by the model in Section 2. The formulation of the problem and our approach to its solution are discussed in Section 3. An algorithm that enables a network of the type defined in Section 2 to offer real-time services is described in Section 4. Finally, Section 5 presents our conclusions and plans for future work.

2. The Network

In its full generality, the network we consider in this study has an arbitrary topology, and may actually result from the interconnection of several networks of various types (LANs and WANs). In it, a message can go from a source host to a destination host passing through a number of intermediate nodes, some of which may be gateways (or routers) connecting one network to another. The functions of a network node may be implemented by hosts or by special purpose communication computers or by both. A source-destination path goes through a number of nodes where the transmitted information can be stored and then forwarded to the next node. Some of the links between adjacent store-and-forward nodes may actually be non-store-and-forward networks, provided their total delay can be bounded. They could be, for example, very high-speed circuit-switched trunks, whose transmission speed is so high that there can be no storing or processing in the switches. The traffic on these trunks will generally include packets from different sources to different destinations, but the delay of each packet will be bounded. Thus, the only assumption we make about the links between store-and-forward nodes is that there be a known and finite bound for the link delay of each packet. Without it, we would not be able to offer hard real-time guarantees. It should be noted that this assumption is not exactly satisfied by links governed by contention-based protocols (e.g., Ethernets, see [Kuro84]), but it is by other types of LANs, such as FDDI rings (see for example [Sevc87], [Dyke88]).

We believe that the model presented above, though not absolutely general, is an adequate characterization of many current and future wide-area networks. How long in time will its validity last is an open question: at some point in the future, store-and-forward nodes might disappear altogether, or some of the links might exhibit unboundable delays (even now, the presence of an Ethernet along the path of a real-time packet raises the specter of invalidity when absolute delay bounds are desired). However, it is hard to predict when the number of networks satisfying our model will become negligible.

A network with these properties could be based on circuit switching, or, since circuit switching is a rather wasteful technique in data communications, on hybrid switching. In the latter case, circuits would be allocated to real-time connections, while non-real-time communications could use the remainder of the bandwidth of each link in packet switching mode. If the boundary between the two portions of a link's total bandwidth is made movable as a function of demand, this solution can exploit network resources more efficiently, but requires fairly complex switching in each node. This in turn increases the difficulty of designing the very-high speed switches that the higher network bandwidths of the future will require.

A solution in which the two types of service (real-time and non-real-time) would be more intimately integrated is one based on packet switching, for example, of the fast packet switching variety mentioned above. But can such a network guarantee performance? This is the main question we plan to answer in this paper, by trying to provide a constructive proof of feasibility.

The first question we have to address is that of the model to be assumed for the switches (or nodes) in the network. For the treatment in this paper, we shall choose a simple node model: a single queueing server with multiple queues, as shown in Figure 1(a). The functions of the

various queues will be explained in the sequel. We shall assume that the service time of each packet in the node server is known to, or easily predicted by, the node.

Another problem to be considered is whether a real-time service can be built at the transport or higher level in the protocol hierarchy on top of a datagram service. We believe that packet delay is much harder to control if each packet going from a given sender to a given receiver can in principle follow any route, and some of these routes may become congested while one or more real-time packets are in transit. In this paper, we therefore plan to restrict our attention to the case in which all packets belonging to a real-time channel always follow the same route. A possible approach, based on source routing, is one in which the source selects the path to each destination and keeps it fixed for the duration of the communication. However, the same result can be achieved by building real-time services on top of a connection-oriented (virtual-circuit) service offered by the network layer. The latter approach will more easily permit reservation of resources in each node along the route, which we believe is necessary to guarantee delay bounds. Even the *flow* abstraction in [Come88] uses a fixed route scheme with resources reserved in advance, in spite of the authors' declared preference for a connectionless approach.

A major drawback of connections is that they have to be established before any communication can take place; this may, on the one hand, require a time longer than the client is able to wait in some real-time applications, and on the other hand make the shipment of a small amount of urgent information by a real-time service slower than by normal datagrams. In the study described here, we have tried to minimize the call setup time by designing an efficient connection establishment scheme.

3. A Formulation of the Problem

As discussed in the previous section, we study the feasibility of supporting performance guarantees in a wide-area network consisting of store-and-forward nodes interconnected by bounded-delay links. The network is assumed to offer connection-oriented service (as well as a connectionless service for those non-real-time communications where this service is preferable) at the network layer.

To formulate the problem in more detail, we introduce the notion of *real-time channel*: a simplex connection between a sender and a receiver that delivers messages (or packets) in sequence and is characterized by performance and reliability parameters. The client (i.e., the entity that requests the establishment of a channel) specifies the values of the parameters to communicate its needs to the service provider, i.e., ultimately, to the network layer. Note that this notion is very close to that of *real-time message stream* presented in [Ande88], except that the latter includes also parameters that are not of the performance or reliability types.

We are concerned here only with the performance-oriented parameters of a channel:

- the channel's capacity, defined as the maximum amount of information that may be outstanding at any given time in the channel;
- the maximum packet size;
- the maximum packet service time in each node;
- the delay bound or bounds for the channel's packets[2];
- the maximum packet loss rate.

[2] Our approach is similar in several respects to the one based on *virtual clocks* described in [Zhan90]. However, there are also many substantial differences, the most macroscopic of which perhaps is the fact that we allow clients to specify any delay bounds they may need along with their throughput bounds (the latter through capacity and packet size

While capacity and size are to be enforced by the client, delay bound and loss rate, whose values are to some extent interdependent [Sumi88], are to be guaranteed by the provider.

There are various types of channels, corresponding to the different types of delay bounds. In [Ande88], the following bounds are considered:

- *deterministic*: the bound D is an absolute one; this is necessary in hard real-time applications;
- *statistical*: the bound is expressed in statistical terms; for instance, the probability that the delay of a packet is smaller than the given bound D must be greater than Z;
- *best-effort*: the bound D is not guaranteed by the provider, which, on the other hand, promises to do its best to satisfy it.

Note that the delay guarantees are assumed to be valid only for those packets that reach the destination. Delay bounds will not apply to packets that are allowed by the given maximum packet loss rate to go undelivered due to buffer overrun or to failures affecting any part of their channel. We also assume that the channels to be established are statistically independent of all the channels sharing totally or partially their route. If there are dependencies between a new channel and some of the existing ones, for instance because the new channel is expected to carry information supplementing that carried by another channel, which must therefore be somehow synchronized with it, the client requesting the creation of the new channel should inform the provider. In this paper, we will not deal with the case in which there are dependencies among channels.

Two important characteristics of the network are to be discussed before describing our approach: flow control and packet scheduling.

First of all, should flow control be window-based or rate-based? Both approaches to flow control seem feasible, but the rate-based one (in which the sender controls its packet sending rate on the basis of its knowledge of the characteristics of the receiver and of those of the channel's path) looks more attractive. Indeed, this solution does not require flow control acknowledgements, which would have to use another simplex channel, and would therefore be quite expensive. Alternatively, they could be sent as datagrams, in which case they may incur a rather large and perhaps highly variable delay. This delay might be too large, especially in long-distance transmissions, to be compatible with the frequencies and regularity of packet generation in most video or audio communication.

Rate-based flow control is feasible because, at channel establishment time, the appropriate resources can be committed, and, in particular, the receiver can check whether it will be able to accept packets at the rate declared by the sender. This verification is made possible by the deadline scheduling policy to be used in the receiver as well as in all the nodes (see below); in fact, the ability of the receiver to handle yet another stream of arriving packets can be verified with the same method (described in Section 4) used to test the ability of each intermediate node.

If flow control is rate-based, it is more convenient to replace the capacity parameter with one or more parameters describing the packet arrival (i.e., input) process. Here, we adopt for this purpose the parameters x_{min}, the minimum packet interarrival time on the channel, and x_{ave}, the minimum value of the average packet interarrival time over an interval of duration I. In other terms, x_{ave} is the average interarrival time during the channel's busiest interval of duration I. The ratio between these two parameters is the simplest possible measure of burstiness for the incoming packet stream. The amount of information ''stored'' in a channel is indeed hard to control on the part of the sender without a sliding-window mechanism, especially in a statistical channel, and even more in a best-effort one; the sender, on the other hand, has full control over, or at least full knowledge of, the packet generation rate, and can make sure that $1/x_{min}$ is never exceeded.

---

parameters), whereas the virtual clock method is only concerned with providing average throughput guarantees.

There is, however, the danger that a malicious user will circumvent any flow control mechanism and send into the network packets at a much higher rate than the declared maximum value, $1/x_{min}$, or maximum average value, $1/x_{ave}$. Indeed, current (and future) distributed systems include personal machines, whose operating system can easily be modified or even replaced by their owner or by an occasional user. The same effect might be caused by a failure in the sending host, even when this is a multi-user, protected-kernel system. If we do not take appropriate countermeasures, such malicious or faulty behavior can prevent the satisfaction of the delay bounds guaranteed to other clients of the real-time service, thereby damaging the clients and destroying the credibility of the service. Thus, a distributed flow control scheme seems absolutely necessary; with the characterization of the input we have selected, the scheme will have to make sure that neither $1/x_{min}$ nor $1/x_{ave}$ are exceeded. One possible solution to this problem is presented in Section 4.

Scheduling in the hosts and in the nodes will be deadline-based. More precisely, we adopt a modification of the EDD (Earliest Due Date) policy that gives, in the case of a conflict, priority to deterministic over statistical channels, and to statistical over best-effort channels. As illustrated in Figure 1, this is not an absolute priority: the packet at the head of the deterministic queue is shipped before the packet at the head of the statistical queue only when the latest possible time for starting its shipment occurs before the statistical packet's deadline; otherwise, the latest possible time for starting the shipment of the statistical packet is compared with the deadline of the first packet in the best-effort queue, and the latter is shipped if its deadline is the earlier of the two times. Only the packets in the last (non-real-time) queue have to wait to be shipped until the other queues are empty; also, their shipment can be preempted, while the others cannot. Note that, as shown in Figure 1, deterministic packets have to be "aligned", i.e., their deadlines must to be suitably decreased if they fall into the last possible interval during which the node can safely process another deterministic packet. Without alignment, the deadlines of deterministic packets could not always be met.

4. An Approach to the Solution

The channel establishment mechanism we propose for the type of network described in Section 2 may be built on top of any procedure that can be used to set up connections. It is reasonable to design the routing algorithm for channel establishment so that it will exploit any available information about the delays along the various possible paths to the destination and about the "real-time load" of neighboring nodes. Besides trying to establish a connection, the mechanism will perform several tests and tentatively reserve resources in each node visited by the establishment request message. In order to make channel establishment fast, we impose on our procedure the restriction that it require only one round trip. Thus, the destination host is the last point along the path where the acceptance/rejection decision for a channel request can be made. When a node is revisited by an establishment message during this message's return trip, the resources previously reserved there must be either committed or released; hence a final, irreversible decision must have already been made.

The tests to be done in each node are concerned with the availability of sufficient bandwidth in the links, as well as processing power and buffer space in the node; that is, with determining whether the new channel can go through the node without jeopardizing the performance guarantees given to the already established channels passing through the same node.

If any test fails at a node, the channel cannot be established along that route; the message will be sent back, either to the sender (which may then decide to wait or try another output link) or to an intermediate node that can try sending the message towards the destination along another

path. When an unsuccessful establishment request message revisits a node on its way back to the source, it frees all the resources that were tentatively reserved there during its forward trip. Thus, acceptance of a channel establishment request cannot be guaranteed. The probability that a request is accepted depends on how close to saturation (in terms of real-time peak load) each node along the route is. This probability cannot be easily estimated, unless the source has a complete and up-to-date knowledge of the node loadings and of the probabilities of the various possible routes. An intelligent routing algorithm could increase the probability of success considerably. What can, instead, be guaranteed is a bound on the response time to a request, though the bound would be provided by the server, not chosen by the client.

If all the tests succeed at all nodes and at the destination host, this host subdivides the delay bound *D* (and the probability *Z* of not exceeding the bound if the channel being established is statistical) among the nodes traversed by the channel, after subtracting the total delays in the links along the route. Let *d* be the maximum delay bound assigned to the channel in a node, and *z* the minimum probability that *d* will not be exceeded in that node. A packet traveling on that channel and arriving at that node at time *t* will usually be assigned a node deadline equal to *t+d*. To satisfy the overall delay bound *D*, it is sufficient (though not necessary) to satisfy the bound *d* (or *d, z*) in each node along the route. For simplicity, we assume that satisfying this sufficient condition is the goal to be met by the establishment scheme in each node.

If all tests succeed, a reply message is sent back to the source host along the channel's route; this message notifies each node about the delay bound that has been assigned to it for the new channel, and commits all the reserved resources. The delay bound assigned to a node is then used to compute the deadline in that node for each packet traveling on that channel (see the calculation of *dl* in Figure 2). When the reply message reaches the source host, this host learns that the requested channel has been set up, and can start using it.

Note that, in the procedure just described, the destination host assigns delay bounds to nodes since we assume that the source host does not know how long the path to the destination will be. If the number of hops were known at the source beforehand, this assignment could be done by the source host, but it would be done less effectively, due to the source's incomplete knowledge of each node's loading situation.

Before discussing the tests to be performed in each node, we have to choose a suitable flow control mechanism. One possible approach, whose execution costs are to be carefully investigated, consists of increasing the deadlines of the "offending" packets (i.e., packets arriving sooner than expected), so that they will be able to go through lightly loaded nodes fairly rapidly, but will be delayed in heavily loaded nodes, where they would seriously interfere with the operation of other channels. When buffer space is limited, some of them might even be dropped because of buffer overflow. Of course, at least some of the buffers will have to be statically allocated to prevent the offending packets from flooding the buffer space of a heavily loaded node and causing packets from other channels to be dropped.

One simple algorithm to increase the deadlines of packets arriving too soon after their predecessors is shown in Figure 2. The algorithm does not need any timers, which may be expensive to maintain. Note that the actual intervals between successive arrivals of a channel's packets at a node may be occasionally shorter than x_{min}, and the averages of the same intervals over *I* occasionally shorter than x_{ave}, either because of a sudden decrease of the load on the previous node along their path or (if the node in question is the first on the path) because of a higher packet rate illegitimately generated by the sending host. However, the distributed flow control algorithm in Figure 2 acts on the packets' deadlines so as to impose on the node the same switching and transmission load as if the packet stream on the channel obeyed the x_{min} and x_{ave} constraints. We can therefore assume that each channel going through the node satisfies those constraints.

To simplify our discussion, we shall assume here that the buffer space available for real-time connections in the network's hosts and nodes is unlimited. This assumption will allow us to

ignore the loss rate parameter, and to postpone the treatment of buffer space allocation and management to a subsequent paper.

When a node receives an establishment request message, it performs two or all three of the following tests:

(a) the *deterministic test*, required only when the channel to be established is deterministic, and involving only the deterministic channels already existing in the node;

(b) the *statistical test*, required for the establishment of both statistical and deterministic channels if at least one statistical channel is present in the node or is to be set up, and involving all deterministic and statistical channels in the node;

(c) the *delay bound test*, which is needed in all cases, and, if successful, is followed by the computation of the minimum delay bound for the channel in the node.

If the request passes the tests, then the node performs a *delay bound computation*, and then sends the establishment message on to the next node.

Note that best-effort channel requests are not subjected to any tests. Hence, they are always accepted. In the sequel, unless explicitly specified, by the term "channel" we shall refer only to deterministic or statistical channels.

The **deterministic test** consists of verifying that enough processing power is available in the node to accommodate the additional deterministic channel without impairing the guarantees given to the others. Since we are dealing with deterministic bounds, this must be true even in the worst possible case, i.e., even when all deterministic channels are sending packets into the node at their maximum rates. The maximum utilization of a node by channel i, whose packets have a maximum service time in the node (see Section 2) equal to t_i, is $t_i/x_{min,i}$. Thus, the condition to be tested is:

$$\sum_j t_j/x_{min,j} \le 1, \tag{1}$$

where the sum extends to all deterministic channels, including the one to be established (channel i).

The **statistical test** has two purposes:

(i) to determine whether for each statistical channel j in the node the probability of a delay higher than the bound d_j is below its maximum tolerable value, $1 - z_j$;

(ii) to provide the destination host with the information that is necessary to compute z_i for the new channel if this is statistical.

Both these purposes can be obtained by computing the *probability of deadline overflow* P_{do}. A packet may be delayed beyond its delay bound in a node because of two reasons: a temporary saturation of the node's processing and switching capacity (*node saturation*), and impossible scheduling constraints (*scheduler saturation*). The latter is avoided, as discussed below (in our description of the delay bound test), whereas the statistical test deals with the former.

A measure of the probability that channel j is active[3] during an interval I is

$$p_j = x_{min,j}/x_{ave,j}. \tag{2}$$

Given m independent channels 1, 2, ... m passing through a node, the probability that the members of a subset C of them are simultaneously active is given by

$$Prob(C) = \prod_{j \in C} p_i \prod_{j \notin C} (1-p_j). \tag{3}$$

[3] The packet arrival pattern on a channel j that minimizes the probability of a deadline overflow is one in which all $I/x_{ave,j}$ packets in I arrive at intervals equal to $x_{min,j}$, while no packets arrive during the remainder of I. In such worst-case situation, channel j is said to be *active* during the periods in which arrivals are occurring. The variable p_j defined in (2) is the fraction of I during which channel j is active.

Let at least one of the channels be statistical. To compute P_{do}, we start by listing all the *overflow combinations*. An overflow combination is a set of channels that, when simultaneously active for a sufficiently long time, will cause packets to miss their deadlines. In other words, an overflow combination is one for which inequality (1) above, with the sum extended to all active channels of both types, is not satisfied.

Let H be the set of overflow combinations in the node, h be a member of H, and $P(h)$ be the probability of occurrence of combination h computed as in (3) above. Then,

$$P_{do} = \sum_{h} P(h). \tag{4}$$

We can then check whether the following inequalities are satisfied:

$$P_{do} \leq 1 - z_j, \tag{5}$$

for each statistical channel j existing in the node. To see whether (5) is satisfied, it is sufficient to verify that

$$P_{do} \leq \min(1-z_j), \text{ or } 1-P_{do} \geq \max z_j. \tag{6}$$

If test (6) is successful, then the value of P_{do} is sent to the destination host for the purpose referred to in (ii) above. The use of this value by that host is discussed below (see (12)). The destination host will try to assign to channel i in the node a value of z_i that satisfies inequality (6).

Figure 3 shows two examples of statistical test. The node being considered has four channels (numbered 1 through 4) already established when it receives a message requesting the establishment of channel 5. This is a deterministic channel whose addition would cause node saturation whenever all channels have maximum rate traffic on them for a sufficiently long time (see Figure 3(a)). Since, as shown in Figure 3(b), the probability of deadline overflow is 0.000161, and its complement to 1 (0.999839) is greater than the z's of the two statistical channels (channels 2 and 3), the node passes the statistical test. The value of P_{do} is not transmitted to the destination host in this case, as channel 5 is deterministic.

The same figure also shows in (a) the characteristics of another new channel, called channel 6, and in (b) the calculation of P_{do} according to equations (3) and (4), under the assumption that channel 5 has been established. The node passes the statistical test also for channel 6, and transmits the value of P_{do} (0.000943) to the destination host, since channel 6 is statistical.

The **delay bound test** determines whether scheduler saturation can be avoided in the node, and, if so, the minimum delay bound to be assigned there to the channel being established so that this goal will be achieved. We have to eliminate this type of saturation for deterministic channels, since we could not provide absolute guarantees otherwise. For statistical channels, we try to avoid it primarily because of the complexity of dealing with it in the general case. A simple example of scheduler saturation is that of two channels with respective node service times 3 and 4 units, and respective node delay bounds 5 and 6 units; if two packets from the two channels arrive simultaneously at the node, the scheduler will in no way be able to schedule them so that their deadlines are met.

To determine whether scheduler saturation is possible in a node, we divide the m (deterministic or statistical) channels passing through the node into two sets: we call A the set of those channels whose delay bound in the node is lower than the sum of all channel service times, and B the set of those channels whose delay bound in the node is greater than or equal to that sum:

$$\text{A}=\{i \mid i=1,\ldots a\,;\, d_i < \sum_{j=1}^{m} t_j\}, \tag{7}$$

$$\text{B}=\{l \mid l=a+1,\ldots m\,;\, d_l \geq \sum_{j=1}^{m} t_j\}, \tag{8}$$

With no loss of generality, we number the a channels in A according to the order in which they would be scheduled by the algorithm in Figure 1 if they arrived all at the same instant: 1 will be the channel that would be shipped first, a the one that would be shipped last.

Let us assume that the node is not saturated: we have already taken into account node saturation, which delays packets beyond their bounds whether or not scheduler saturation is present, in the statistical test; also, let us assume for simplicity of proof that

$$x_{min,l} \geq \sum_{j=1}^{m} t_j \tag{9}$$

for l=1,...m. This assumption is equivalent to postulating that no subsequent arrivals of packets at the node will interfere with the transmission of those (one per active channel) we are considering. A discussion of how to proceed when the assumption is not satisfied can be found in the Appendix. The following theorem, on which we will base the delay bound test, holds.

Theorem 1: Scheduler saturation is impossible if and only if