BUFFER SPACE ALLOCATION FOR REAL-TIME CHANNELS IN A PACKET-SWITCHING NETWORK

Domenico Ferrari and Dinesh C. Verma
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
and International Computer Science Institute
Berkeley, California, U.S.A.

Abstract

Broadband integrated networks will have to offer *real-time communication services*; that is, they will have to transport information with performance guarantees. A paper previously published by the authors presented a scheme for establishing real-time channels in a pure packet-switching network; that scheme did not include any method for allocating buffer space in the network's nodes to the channels being established. This paper completes the description and evaluation of that scheme, since it presents one such method, and some of the results of the extensive simulations performed to test it. The method is found to be correct and to have a low overhead. While the utilization of the buffer space allocated to the statistical channels is often quite low, thereby indicating that our worst-case approach tends to overallocate space to those channels, the space our method gives to deterministic channels seems to be reasonably well utilized.

This research has been supported in part by AT&T Bell Laboratories, Hitachi, Ltd., the University of California under a MICRO grant, the National Science Foundation under Grant No. CDA-8722788, and the International Computer Science Institute. The views and conclusions in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of any of the sponsoring organizations.

1. Introduction

Broadband integrated networks will have to offer *real-time communication services*, i.e., to transport information with performance guarantees, primarily to satisfy the requirements of digital audio and digital video [Lein89]. The specific performance guarantees that will be needed will vary with the type of traffic (video, audio, or their various combinations with each other and with other information media) as well as with the application [Ferr90]. Once such services become available, they are likely to be used even for types of traffic that do not include video or audio, for example, alarm conditions in distributed real-time systems, urgent remote database queries, or file transfers requiring lower bounds on throughput. Thus, real-time communication services are going to be essential in the integrated networks of the future, along with the more traditional, non-real-time ones such as datagram and virtual circuit services of various kinds.

Three basic approaches can be used to provide real-time communication: circuit switching, packet switching, and hybrid switching. A *circuit-switching* network will allocate its resources continuously and exclusively to each connection for the entire duration of the connection [Harr80]. With *packet switching*, resource allocation is neither continuous nor exclusive; that is, resources are shared among various conversations on a packet-by-packet basis, and the only privileges certain conversations may receive consist of a higher priority (preemptive, but with a non-preemptable granule, i.e., the packet) in the use of shared resources. In a *hybrid-switching* network, the two previously described approaches coexist: certain conversations (typically, those needing performance guarantees) are carried by the circuit-switching portions, others by the packet-switching portion of the network.

If a sufficient amount of resources is allocated to a connection in a circuit-switching network, the given performance guarantees can be satisfied. However, bursty traffic will make a poor use of these dedicated resources. A hybrid solution could solve this problem if real-time traffic were non-bursty, and if bursty traffic did not need performance guarantees; this, unfortunately, is not the case, since compressed video and audio are bursty, and since other bursty traffic is likely to require real-time services. It is therefore useful to investigate the question whether packet switching can be used to provide performance guarantees, and, if so, whether and under what conditions a packet-switching solution is more convenient than the other two.

In [Ferr89] and [FeVe90], we have provided an answer to the question of feasibility by proposing and evaluating a scheme on which the design of a real-time service for an integrated wide-area packet-switching network could be based. One of the assumptions made in that paper was that the amount of buffer space available in each node would be sufficient to hold all the packets waiting to be forwarded to the next node, even in the worst case. In this paper, we eliminate that assumption, and describe and evaluate an algorithm for determining the amount of buffer space that should be allocated to each real-time connection in each node on the basis of the characteristics of the traffic on that connection and the bound on the probability of packet loss specified by the client. The algorithm completes the scheme described in [Ferr89] and [FeVe90], and the constructive proof of feasibility that scheme was intended to provide.

The important features of the scheme (under the infinite buffer space assumption) are summarized in Section 2. Section 3 presents the buffer space allocation algorithm. The simulation experiments we have run to evaluate the algorithm and their results are discussed in Section 4. Finally, Section 5 provides a summary of our study and suggestions for future work.

Related work can be found in [CoYa88], [Zhan89] and [OhON88].

2. A Scheme for Providing Real-Time Services in a Packet-Switching Network

The solution proposed in [Ferr89] and [FeVe90] is based on the notion of *real-time channel*, a simplex connection over which packets can travel from the sender to the receiver with performance satisfying given guarantees. Our scheme allows the following performance guarantees to be offered to the clients of the service:

(1) *throughput bounds*: both
  (i) minimum peak throughput ($\theta_{\min}$) to be provided by the network, and
  (ii) minimum average throughput (θ_{ave}, I), the lower bound of the throughput averaged over intervals of duration I;

(2) *delay bounds*: either
  (i) deterministic maximum delay ($D_{\max}$): $D_i \le D_{\max}$ for all i; or
  (ii) statistical maximum delay ($D_{max}, Z_{\min}$): $Prob\,(D \le D_{\max}) \ge Z_{\min}$.

The desired throughput bounds are specified implicitly in the client's characterization of the traffic to be carried by the real-time channel whose establishment the client is requesting. Calling "arrival time" the time a packet enters the network though the sender's interface, this characterization consists of the following parameters:

(a) the minimum packet interarrival time $x_{\min}$;
(b) the minimum average packet interarrival time x_{ave};
(c) the duration I of the time interval over which x_{ave} is computed; x_{ave} is the smallest of the averages, computed over all of the intervals of duration I, of packet interarrival times;
(d) the maximum packet size $s_{\max}$;
(e) the service time t of a packet in each node along the channel's route.

Note that (d) and (e) are likely to be known to the network, and may not have to be specified by the client. Note also that requiring the client to specify $x_{\min}$ and x_{ave} implies a pledge on the client's part to keep the minimum and average intervals between consecutive packets at the channel's entrance to the network larger than or equal to $x_{\min}$ and x_{ave} respectively. The service provider must, however, ensure and verify that the client honors this pledge at all times, otherwise the performance given to other clients may be in danger. The solution to this problem adopted in [Ferr89] and [FeVe90] consists of having each node compute the minimum and average intervals, and suitably increase the local deadlines of those packets that do not satisfy the client's pledge. This *distributed rate control* algorithm is not only executed in the edge nodes, but in all nodes, since load fluctuations can indeed cause local and temporary violations even if all clients scrupulously honor their pledges.

The network we consider is a general-topology, point-to-point, store-and-forward, packet-switching one. The links are assumed to be bounded-delay: that is, the delay incurred by a packet over a link must have a finite and known upper bound. This requirement is satisfied by links consisting of pure communication channels, whose delays are those due to signal propagation, as well as by a number of more complicated structures, such as entire networks, but not by all of them; for instance, if the link connecting two nodes is an Ethernet, the bounded-delay requirement is not exactly satisfied.

Each node is assumed to be suitably modeled as a single server (in the queueing theoretical sense of the term "server") with multiple queues and multi-class deadline-based scheduling [Ferr89]. The node scheduler handles at least three queues, listed here in order of decreasing priority: the deterministic queue, where packets traveling on a real-time channel with deterministic delay guarantees are stored; the statistical queue, for packets on real-time channels with statistical guarantees; and a queue for everything else that is to be processed by the node. Each queue is ordered according to packet deadlines in the node. When the node is free to start shipping a new packet, the deadline of the packet at the head of the statistical queue is compared with the time at which the packet at the head of the deterministic queue should start being handled by the node for its local deadline to be met (we shall call this the "start deadline", to distinguish it from the "end deadline", or simply "deadline"). If the former is larger than the latter, the deterministic packet is shipped; if it is smaller, the statistical one takes precedence. Packets in the third queue are shipped only when the other two queues are empty. The algorithm can be generalized in the obvious way to the case of q queues, $q-1$ of which are for real-time packets with different priorities.

That of real-time channel is a logical concept: its physical realization could in principle be of various types, including multiple routes or datagram service between the sender and the receiver. However, obtaining performance guarantees and sequenced delivery (an important requirement in video and audio communications) is either very hard or impossible if a real-time channel is not realized as a connection with a single, fixed physical route. Since reservation of resources is necessary (not in the circuit-switching sense of this term, but to keep track of the potential real-time load on each resource and to prevent this load from exceeding its bounds), real-time channels are to be established explicitly when they are requested by a client (unless an existing one can be use for that client's traffic). The establishment procedure consists of selecting a route in a way analogous to that for a virtual circuit, though the routing algorithm might somehow be influenced by the knowledge that the channel to be set up is a guaranteed-performance one. However, besides selecting the route, the procedure will include computations and tests in each node along the route, the tentative reservation of node resources, and the transmission of information from each node to the receiver.

If the tests in a node are all successful, the *establishment message* is sent to the next node selected by the routing algorithm, and so on until it reaches the receiver. The receiver is the final decision-maker, i.e., it runs the last tests; if these too are successful, then the request is accepted, and the receiver sends an *acceptance message* back to the sender; this message transmits information from the receiver to each node on the route, and commits the resources (or the fractions of them that are really needed) tentatively reserved by the establishment message. If, on the other hand, one of the tests is unsuccessful, either in a node or in the receiver, the establish message stops there, and a *rejection message* is sent back to the source. This message undoes the partial, tentatively established connection, and releases the tentatively reserved resources in each node along that connection.

The choice of the receiver as the final decision point was dictated by the desirability of a fast establishment procedure. Indeed, our procedure requires a single round-trip; it can, however, be made even faster by optimistically starting packet shipments before the new channel has been completely established [DaVe89].

If the request for the creation of a new real-time channel is accepted, each node along its route has to keep information about the channel; some of this information is to be used for the correct handling of the packets traveling on the channel, some other during the tests for the establishment of new channels through the node, and some for both purposes. Besides storing routing information and all the channel's parameters specified by the client (e.g., $x_{\min}$, x_{ave}, $s_{\max}$, and so on), the node must be given and keep the values of the following parameters:

(a) the *delay bound* $d_{i,n}$ of channel i in node n; this is the maximum (in a deterministic or statistical sense, depending on channel i's type) time a packet on channel i should spend in the node; the *deadline* of a packet arriving at the node at time t will be computed as $t+d_{i,n}$, unless the rate control algorithm (see Section 3) intervenes to modify it;

(b) the *delay bound compliance probability* $z_{i,n}$ of statistical channel i in node n; this is the minimum probability that a packet on channel i will not be delayed more than $d_{i,n}$ in node n.

The values of $d_{i,n}$ and $z_{i,n}$ are computed by the receiver, the first entity, among those involved in a channel's establishment, that knows all the necessary information to do these computations. While $d_{i,n}$ is used both to determine packet deadlines and in the delay bound test for the establishment of a new channel (see below), $z_{i,n}$ only appears in establishment tests (see the statistical test below). Since the network has an arbitrary topology, each node must have its own independent and local characterization of the guarantees it has given each channel passing through it. This role is played by $d_{i,n}$ and $z_{i,n}$, and is the only justification for the latter parameter's existence. For the same reason, the value of $Z_{min,i}$ specified by the client is factored by the receiver into the individual node compliance probabilities $z_{i,n}$:

$$Z_{min,i} = \prod_{n=1}^{N} z_{i,n} \ , \tag{1}$$

where N is the total number of nodes along the channel's route. This implies the pessimistic assumption that a packet delayed beyond its local bound in a node will not be able to satisfy the channel's global delay bound.

The tests to be performed in each node are as follows (see [FeVe90]).

(1) *The deterministic test.* This test is to be performed for any deterministic channel to be established, and involves the deterministic channels already passing through the node. It verifies that the node has enough processing power to handle the additional deterministic channel without impairing the guarantees given to the other channels in the node; the power is sufficient if the utilization of the node's processor due to deterministic channels is less than 1 even after the addition of the new channel and even in the worst case. The computation requires knowledge of t and $x_{\min}$ for each of the channels involved. Note that we can assume that $x_{\min}$ and x_{ave} remain the interpacket time bounds along the entire route of the channel because of our rate control policy (see Section 3).

(2) *The statistical test.* This is to be performed whenever at least one statistical channel is involved, i.e., either to be established or already established in the node. It determines whether, for each statistical channel j already passing through node n, the probability P_{do} that its packets will incur a delay higher than its bound $d_{j,n}$ in the node is below the maximum tolerable value for that probability, $1-z_{j,n}$. P_{do} is assumed to be the same for all packets on all channels passing through the node, and is computed from the probabilities of all the combinations of active channels that can potentially saturate the node. This computation assumes that channel traffic patterns are statistically independent. The value of

P_{do} thus calculated is transmitted to the receiver via the establishment message, to be used in the computation of $z_{i,n}$.

(3) *The delay bound test*. This test is performed in all cases, to determine whether *scheduler saturation* can be avoided in the node even after the creation of the new channel. Scheduler saturation makes it impossible for the node scheduler to satisfy all delay bounds in spite of the absence of *node saturation*, which is handled by the statistical test. This condition is caused by delay bounds that are too small compared with the service times t and the minimum interpacket times $x_{\min}$. To perform the delay bound test, we compute the minimum delay bound $d^{l}_{i,n}$ that can be assigned to the new channel in the node. This computation requires the knowledge of $x_{min,j}$, $t_{j,n}$, $d_{j,n}$, $x_{min,i}$, and $t_{i,n}$ $(j \neq i)$. If the new channel passes the test, the value of $d^{l}_{i,n}$ is transmitted to the receiver via the establishment message.

The receiver performs two tests and two computations to determine the values of $d_{i,n}$ and $z_{i,n}$, which it then sends to all the nodes on the route via the acceptance message. If the channel to be established is deterministic, only one test and one computation (that of the $d_{i,n}$'s) are needed. The tests are called the D test and the Z test.

(1) *The D test*. This test verifies that the sum of the $d^{l}_{i,n}$'s over the channel's route is smaller than or equal to $D^{*}_{\max} = D_{\max} - \sum link\ delays$.

(2) *The Z test*. This test consists of verifying that the combination of all delay bound compliance probabilities will be greater than or equal to $Z_{min,i}$:

$$\prod_{n=1}^{N} (1 - P_{do,n}) \geq Z_{min,i} . \tag{2}$$

There are various possible approaches to the computation of $d_{i,n}$ and $z_{i,n}$. One of them, for example, is described in [FeVe90].

3. A Buffer Space Allocation Algorithm

In [FeVe90], another client requirement the real-time channel establishment scheme could satisfy, besides the throughput and delay bounds defined in Section 2, was a *reliability bound* ($W_{\min}$):

$$Prob\,(packet\ sent\ is\ delivered\ correctly) \geq W_{\min}. \tag{3}$$

There are two possible causes of packet loss:

(i) transmission error, usually due to noise or failure of some network component, and

(ii) buffer overrun, usually due to a space shortage in a node of the network or in the receiver.

While problem (ii) causes packets to be really lost, i.e., to go undelivered, transmission errors often garble them. In real-time wide-area communication, the use of such error control mechanisms as acknowledgements and retransmission is highly questionable, since the long delays inherent in the operation of these mechanisms are likely to prevent the packets eventually delivered correctly from satisfying their delay bounds, hence from being usable by the receiver. Thus, we can, at least in most applications, consider as lost any packet that is delivered incorrectly.

Of all the packets that are shipped on a real-time channel, at least $100W_{\min}$% must be delivered correctly. If the channel is statistical, at least $100W_{\min}$% of those delivered correctly must be delivered on time. Thus, the packets that are delivered correctly and on time will be at least $100W_{\min}Z_{\min}$% of those that were shipped by the sender. Some clients may be more interested in $W_{\min}Z_{\min}$ than in the individual values of these two bounds, and may want to leave the decision about what these values should be to the real-time server [Ferr90].

We assume that the network is characterized by a known error rate, and that the given value of $W_{\min}$ will be divided by the complement to 1 of this error rate to obtain the lower bound for the probability of buffer overrun. To avoid complicating our symbology, we shall use $W_{\min}$ in the sequel to denote not just the correct-delivery probability bound, but one of its two factors, namely, the lower bound of the probability that no buffer will overflow, thereby causing packet loss.

In the scheme presented in [FeVe90], the probability of packet loss was always 0, since we assumed no limitations on the amount of buffer space allocated to each channel in each node. In this paper, we remove that assumption, and propose and evaluate an algorithm that assigns reasonable amounts of buffer space to channels taking into account the given value of $W_{\min}$. This value will be factored by the receiver into node contributions $w_{i,n}$ as is done for $Z_{\min}$ and because of the same reasons explained in Section 2:

$$W_{min,i} = \prod_{n=1}^{N} w_{i,n} . \tag{4}$$

The implication of this expression, that is, that a packet lost in node n is lost forever, is of course true. The treatment of the problem for statistical channels is quite different from (and much more complex than) that for deterministic channels. After examining both, we briefly describe an algorithm, based on these treatments, for computing the space to be allocated in each node to a new channel being established.

3.1. Buffer space for a deterministic channel

Let us assume $w_{i,n} = 1$ for channel i in node n. This means that the client has requested a guarantee that there will be no packets lost because of buffer overruns on that channel. The scheme described in [FeVe90] guarantees that in node n, if channel i is deterministic, each of its packets will spend a time not longer than $d_{i,n}$. In the worst case, a packet on channel i will arrive at the node at intervals of length $x_{min,i}$. Assuming the residence time of each packet in node n to be $d_{i,n}$, which is clearly a worst-case assumption, the amount of space in bytes needed by the channel in that node will be

$$SPACE(i,n) = s_{max,i} \left\lceil \frac{d_{i,n}}{x_{min,i}} \right\rceil . \tag{5}$$

If we now assume $w_{i,n} < 1$, we may be able to reduce $SPACE(i,n)$ by up to u packet buffers. In the worst case, subtracting u buffers from the $b = SPACE(i,n)/s_{max,i}$ buffers that are needed to avoid losses may cause up to u packets out of every b to be dropped. Thus, we will have

$$1 - w_{i,n} = \frac{u}{b}, \tag{6}$$

hence the number of packet buffers needed will be

$$b' = \lceil b - u \rceil = \left\lceil b w_{i,n} \right\rceil , \tag{7}$$

and the amount of buffer space in bytes

$$SPACE(i,n) = s_{max,i} \left\lceil \frac{d_{i,n}}{x_{min,i}} w_{i,n} \right\rceil . \tag{8}$$

3.2. Buffer space for a statistical channel

We begin by assuming again $w_{i,n} = 1$. The scheme described in [FeVe90] is such that, if node n is not saturated (i.e., if $\sum (t_{i,n} / x_{min,i}) = 1$), all packets carried by any statistical channel satisfy that channel's delay bound $d_{i,n}$. Thus, a non-saturated node treats deterministic and statistical channels in the same way (though the actual delays incurred by the former are shorter because of the scheduler's preference for them). When, on the other hand, the node is saturated, some of the statistical packets may be delayed beyond their bound and have therefore to stay longer in the buffers. In saturation, expressions (5) and (8) no longer apply.

Let $d_{i,n}^{sat}$ be the maximum delay bound in node n for channel i in a particular saturated situation. It should be noted that to each such situation there will generally correspond a different value of this bound; however, given certain channels established in node n, with $\sum (t_{i,n} / s_{min,i}) > 1$, there will be a maximum packet residence time in the node, and any value greater than or equal to this time will be a valid one for $d_{i,n}^{sat}$.

The maximum time spent by a channel i packet in node n can be computed by observing that the worst case will occur at the end of the longest possible interval during which all channels are simultaneously active and are sending packets at their fastest rates. Let t_a be an instant at the beginning of this interval when a packet on channel i arrives at node n. Since a relatively long time usually elapses between the instant a node becomes saturated and the first instant at which a delay bound is violated, we may assume that this packet leaves the node at a time not later than $t_a + d_{i,n}$. Let now t_b be the arrival time of the channel i packet closest to the end of the saturation interval being considered; this packet will leave the node at time $t_b + d_{i,n}^{sat}$, and the node will be busy without interruptions at least between $t_a + d_{i,n}$ and $t_b + d_{i,n}^{sat}$.

The packets that cause this activity in the node are those whose (start or end) deadlines are greater than $t_a + d_{i,n}$ and smaller than $t_b + d_{i,n}$. During an interval of duration $t_b - t_a$, the number of these packets will equal the total number of arrivals, and each arrival on a channel j will contribute a term $t_{j,n}$ to the node's busy time. Thus, we can write

$$t_b + d_{i,n}^{sat} - t_a - d_{i,n} = \sum_j \left\lceil \frac{t_b - t_a}{x_{min,j}} \right\rceil t_{j,n} , \tag{9}$$

from which, ignoring the ceiling and noting that the portion of the worst interval of duration I in which node n is saturated has length $I \; P_{do}$, we obtain

$$d_{i,n}^{sat} = d_{i,n} + I \; P_{do} \left(\sum_j \frac{t_{j,n}}{x_{min,j}} - 1 \right). \tag{10}$$

To calculate $SPACE(i,n)$ when n is saturated (i.e., when $\sum (t_{j,n} / x_{min,j}) > 1$), we replace $d_{i,n}$

in (5) with $d_{i,n}^{sat}$ as given by (10). When n is not saturated, expression (5) will suffice even for a statistical channel.

If $w_{i,n} < 1$, we can use (8) in a non-saturated node, and

$$SPACE(i,n) = s_{max,i} \left\lceil \frac{d_{i,n}^{sat}}{x_{min,i}} w_{i,n} \right\rceil \tag{11}$$

in a saturated node.

3.3. The algorithm

If node n is not saturated, the amount of buffer space a real-time channel i to be established needs in n can be determined using equations (5) or (8), depending on whether $w_{i,n} = 1$ or $w_{i,n} < 1$. If i has a statistical delay bound and n is saturated, then we must use either equation (5) with $d_{i,n}^{sat}$ (given by (10)) in lieu of $d_{i,n}$ or equation (11), again depending on whether $w_{i,n} = 1$ or $w_{i,n} < 1$.

In all cases, we need to know the value of $d_{i,n}$. Since during the forward trip of the establishment message this value is not known, we have to estimate it in order to determine whether the node has sufficient buffer space for the new channel, and, if so, to reserve that space until an acceptance or rejection message concerning channel i is received by the node. Obviously, since we cannot compute $d_{i,n}$ exactly, we have to overestimate it. A simple, though very gross, estimate satisfying this condition is the value of the channel's delay bound $D_{max,i}$; of course, better (but more expensive) estimates can also be easily devised. An acceptance message carries within itself the value of $d_{i,n}$, which is used by the node to compute the amount of buffer space to be permanently reserved for channel i; if it was grossly overestimated, this amount will be substantially smaller than the one that was set aside during the forward trip.

If $W_{min,i} < 1$, it is likely that we will have $w_{i,n} < 1$. Since the exact value of $w_{i,n}$ is not known during the forward trip, we will set $w_{i,n} = 1$ in (8) and (11) at that time; in other words, we will always use (5) or its version with $d_{i,n}^{sat}$ during the forward trip, and switch to either (8) or (11) if and when an acceptance message is received by the node.

If node n is saturated and channel i is statistical, we use (10) to compute $d^{i,n^{sat}}$. The term $\sum t_{j,n}/x_{min,j}$, which must be computed in any case (even if the deterministic test is not needed) to determine whether n is in saturation, includes the contribution of channel i. The value of P_{do} is known from the statistical test computation, which must therefore be performed before the *buffer space test*. Again, we do not know $d_{i,n}$, and we have to estimate it, for example by setting it equal to $D_{max,i}$ during the forward trip.

Since every new channel established through the node increases the value of $d_{i,n}^{sat}$ for each of the channels already traversing the node, $SPACE(j,n)$ must be computed at every new request for each channel j when and until the node is saturated, and will normally grow. If the space allocated to channel j was equal to the value of $SPACE(j,n)$ calculated when the node was not saturated, then either we are allowed to increase space allocations or we have to reject the request. Thus, if space increases for existing channels are to be excluded, the initial allocation should be greater than $SPACE(j,n)$, otherwise the potential for a better exploitation of the network's resources by statistical channels will be severely curtailed.

If channel space allocations are allowed to grow, it would seem useful to reduce them when existing channels in the node are deleted. However, this is usually not a wise strategy, as the space freed by the deleted channels is likely to suffice for the subsequent creation of new channels, whose number is going to be limited by the same admission control mechanisms that have been operating since the beginning. In other terms, the buffer space a channel has obtained up to that point is a better estimate of its needs at that level of node saturation than any of the previous allocations. The only persuasive reason for reducing the sizes of space allocations would be to give some more space to non-real-time traffic if this has been squeezed too much by the growth of the real-time component. Note that this would be a symptom of a design flaw, i.e., of insufficient total buffer space in the node.

After the amount by which the allocations should grow has been determined, the buffer space test simply consists of checking whether there is enough free space for this growth. Within limits, this "free" space may come from the pool that is being used by non-real-time traffic.

The receiver, together with the values of $d_{i,n}$ and $z_{i,n}$, computes those of the $w_{i,n}$'s, for example as follows:

$$w_{i,n} = W_{min,i}^{1/N}, \tag{12}$$

where N is the total number of nodes along the channel's route. A better approach is one that does not reject a request in a node n if the space available is smaller than the estimated value of $SPACE(i,n)$, but introduces the notion of *offered buffer space*, defined as $B_{o,n} = \min(\textit{available space}, SPACE(i,n))$. If we set

$$b_{i,n} = \frac{B_{o,n}}{SPACE(i,n)}$$