A Dynamic Bandwidth Allocation Mechanism for Slotted Ring Networks*

L. OROZCO-BARBOSA
University of Ottawa, School of Information Technology and Engineering
161 Louis Pasteur, St. Ottawa, ON, K1N 6N5 CANADA

Abstract - The Connection Oriented Data Service (CO service) has been defined by the IEEE 802.6 committee to support variable-bit-rate (VBR) data services over DQDB MANs. The main aim of the CO service is to be able to guarantee the QoS required by a wide variety of VBR services. In this paper, we derive an adaptive bandwidth allocation control mechanism for the DQDB CO Data service. Numerical results demonstrate the effectiveness of the proposed mechanism in providing the guarantees in terms of the throughput required by the applications under changing network conditions.

Key-Words: Computer Networks, MAC Protocols, MANs

1 Introduction

DQDB (distributed queue dual bus) has been selected by the working group IEEE 802.6 as the standard for metropolitan area networks (MANs) [1]. DQDB has been widely studied. A bibliography listing 171 papers about DQDB is given in [2]. The CO service of DQDB has been designed around a control mechanism known as the *guaranteed bandwidth* (GBW) protocol [3]. Several studies have been conducted on the performance of the GBW protocol being used as the underlying mechanism of the CO service of DQDB MANs [4][5]. However, none of these studies has clearly specified the requirements for the effective implementation of the GBW protocol. In this paper, we undertake a detailed study of the implementation issues towards this end.

From now on, we will refer to the connection-oriented data service simply as the CO service and to the connectionless service as the QA service. We have preferred to keep the name QA service since this term has been widely used in the literature. For simplicity, we will refer to the nodes implementing the CO and QA services simply by CO nodes and QA nodes, respectively. Furthermore, we will often refer to node 1 as the node placed at the upstream end of the data bus and all subsequent nodes as nodes *i*, for *i* = 2, 3, ... *n* with node *n* being the one placed at the downstream end of the data bus [1][3].

Suppose a cycle length of *N* time slots and let *r* = 1,2,..., *N* denote the enumeration of the slots from left to right. Assuming that there are *N* slots in each cycle (frame) and that there are *L* cycles for the period under consideration.

* This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant number RGPIN121600. The author wishes to acknowledge the fruitful discussion with N. Ahmed during the early stages of this research.

2. Formal Model

In this section, we formally describe the operation of the DQDB MAC protocol. This description forms the basis of our model. As already stated, a DQDB network consists of two contra-buses. In order to control the access to the busses, each node comprises two counters. The request counter (RC) and the count down (CD) counter. These two counters are updated in a slot-time basis according to the following rules:

$$RC^u_{i,k}(r) = \{(RC^u_{i,k}(r-1) + \mathrm{I}(R^l_{i,k}(r)=1) - \mathrm{I}(B^u_{i,k}(r)=0)) \vee \{0\}\} \bullet \{1-\mathrm{I}(R^l_{i,k}(r)=0, CD^u_{i,k}(r-1)=0, Q^u_{i,k}(r-1)\geq 1)\} \quad (1)$$

for $r = 1,2,\ldots,N-1$, $k = 1,2,\ldots,K$

where *RC(x)* denotes the value of the request counter at slot-time *x*, *R(x)* is the status of the request bit of the slot passing by station *i* at the *k* cycle on the lower bus at slot-time *x*. Similarly, *B(x)* represents the status of the busy bit of the slot passing by station *i* at the *k* cycle, slot-time *x* on the upper bus. **I** is the indicator function and $(A \vee B)$ indicates the maximum of A and B. *Q(x)* is the number of packets queued in the transmit queue of the *i* station. According to this notation, the expression reads as follows: the value of *RC* at slot-time *r* is equal to the minimum of zero and the value of the *RC* at the previous slot-time incremented by one if a new request at the beginning of the current slot-time is received or decremented by one if a free slot passes by the station. While the second line indicates that the *RC* counter is reset to zero if the request bit is zero, the count down counter in the previous slot-time is zero, meaning that there is no previous request remaining ungranted, and the node has a segment to send.

Similarly, the operation of the *CD* counter can be specified as follows:

$$CD^{u}_{i,k}(j+r-1) = \{(RC^{u}_{i,k}(r-1) + \mathrm{I}R^{l}_{i,k}(r)=1) - \mathrm{I}\left(B^{u}_{i,k}(r)=0\right) \vee 0) \bullet \mathrm{I}(R^{l}_{i,k}(r)=0, CD^{u}_{i,k}(r-1)=0, Q^{u}_{i,k}(r-1) \geq 1\} - \sum_{s=r+1}^{j+r-1} \mathrm{I}(B^{u}_{i,k}(r+s)=0) \vee \{0\} \tag{2}$$

for $j = 1,2 \ldots N-r, r = 1,2 \ldots N-1, k = 1,2 \ldots L$

This expression reads as follows. The value of the *CD* counter of the *i* station for the upper bus at the *k* cycle and slot-time $j + r - 1$ is given by the maximum of the value of the *RC* counter at slot-time $r - 1$ decremented each time a free slot passes by the station via the upper bus. The time at which the value of the RC is transferred to the *CD* counter is given by slot-time $r - 1$. This corresponds to the slot-time when the station is able to place the request. In other words, the count of free slots is started from the slot-time following the placement of the request at slot-time $r - 1$ as given by (1). The count down of the *CD* counter stops when this one reaches zero, this is indicated by the fact that the *CD* cannot be less than zero.

3. The GBW Protocol

In [4], Karvelas has described the operation of the Guaranteed Bandwidth (GBW) protocol. Under the GBW protocol, a Credit Counter (*CC*) and three parameters: the Segment Cost (*SGC*), the Income Per Slot (*INC*), and the maximum credit (*CRM*) are used by each node supporting the CO data service. A node increases its *CC* by *INC* for every slot passing by on the reverse bus, also called the request bus, i.e., the bus used by the node to set a request for a free slot on the data bus. Whenever the node has a segment ready to transmit for which a request has not been already sent and if *CC* > *SGC*, the node issues a request and decreases *CC* by *SGC*. By requiring each node to transmit only on reserved slots and by appropriately selecting the values of *INC* and *SGC*, the GBW protocol can guarantee the bandwidth required by a CO node. For instance, if the channel bandwidth is 155 Mbps by setting *INC*=30 and *SGC*=155 a bandwidth of 30 Mbps is guaranteed. This can also be explained as follows. Assume that *CC* has been set initially to 0. *CC* will be increased by *INC* for every slot passing by on the reverse bus. *CC* will only become greater than *SGC* after having seen 6 slots on the reverse bus. At this time the node can issue a second request while setting *CC*=25. The node is not allowed to place another request but 5 slots after while setting *CC* to 20. Following this procedure, *CC* will be reset to 0 again (one period), 31 slots from the initial time during which a total of 6 requests have been placed. Therefore, in one period, 6 out of 31 slots will be used by the node. In the case of a channel rate of 155 Mbps, the guaranteed bandwidth is 155 x 6/31=30 Mbps.

In order to prevent *CC* for increasing indefinitely during the periods when the node is idle, a maximum value for the accumulated credit, *CRM*, is introduced. That is, if the value of *CC* exceeds *CRM*, the node will not continue to increase *CC* although it will observe slots passing by on the reverse bus. According to [4] the minimum value of *CRM* that will provide the node with the guaranteed throughput is the one that satisfies the following inequality:

$$CRM \geq \lceil SGC/INC \rceil \bullet INC \tag{3}$$

where $\lceil x \rceil$ is the minimum integer equal or greater than *x*.

The GBW protocol should be used by the nodes implementing the CO service. Even though some performance studies have been carried out to evaluate the effectiveness of the GBW [4][5], setting up CRM according to (3) does not always guarantee providing the guaranteed bandwidth.

4. The DQDB CO Data Service

Unlike the QA service, a CO node is required to send a request for each segment to transmit and keep track of the position of all the segments already admitted into its local reservation queue. The QA service can send its segment as soon as its *CD* reaches zero, even if the node has not been able to actually place the corresponding request. In the case of the CO service, the node must first place a request before actually transmitting a segment. This requirement is important for the proper operation of the CO service. To implement this feature, we suggest the use of a set of counters, equivalent in operation to the *RC* and *CD* counters used by the QA service [1]. This will allow each CO node to record the downstream CO service requests and let certain number of free slots go in order to be used by downstream CO nodes. Since a CO node can send multiple request, for each segment request, a *CD* is needed to hold the number of downstream CO request sent before it. Therefore, an array of *CD* is required. The number of *CD* needed depends on the bandwidth requirements of the CO node and the distance between the CO node and the farthest upstream node. This value (n_{CD}) is bounded by the following formula:

$$n_{CD} \leq \lceil 2d_i \bullet B_i / B_t \rceil \tag{4}$$

where d_i is the distance between CO node i and the most upstream node, B_i is the required bandwidth of CO node i and B_t is the total bandwidth of DQDB network.

To demonstrate the effectiveness of this implementation, we simulated a system consisting of three nodes with the most upstream node, node 1, implementing the QA service while the two other nodes, node 2 and node 3, implement the CO service. At the beginning of the simulation, the QA node occupies the whole channel capacity. It is further assumed that each one of the two CO nodes requires 25% of the overall channel capacity. Under the considered scenario, node 2 becomes active before node 3. When node 3 becomes active, node 2 remains active and ends its transmission after node 3 has finished transmitting. Figure 1 shows the access delay for nodes 2 and 3. The access delay is defined as the time elapsed between the instant when a node places a request and the instant when the node initiates the transmission of the segment associated to the request. As shown in the figure, when the node 2 becomes active and as long as node 3 remains inactive, the access delay for node 2 remains constant to 20 slots. This delay corresponds to the round-trip delay between node 2 and 1. After node 3 starts to send its segments, the access delay of node 2 remains at 20 slots while node 3 experiences an access delay of 40 slots. These results are in close agreement with the requirements of the DQDB CO data service as specified in [3].

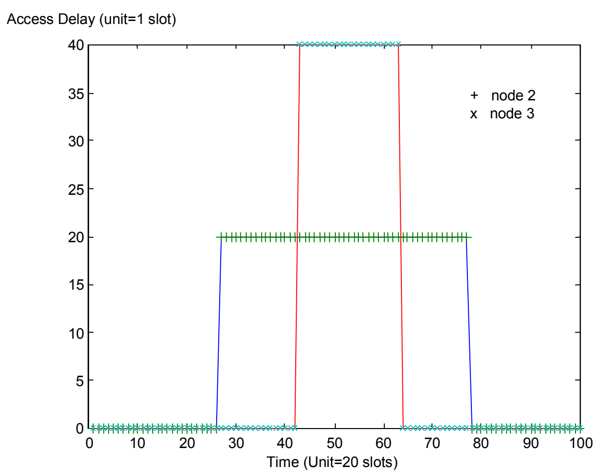

Fig. 1 Access delay

5. An Adaptive GBW Mechanism

In this section, we propose the use of a method to dynamically set up the CO service parameters as required to guarantee the QoS requirements of various applications.

Assume a DQDB network with the most upstream node (node 1) implementing the QA service and two other nodes (node 2 and 3) implementing the CO service and requiring 25% and 40% of the total bandwidth, respectively. The distance between two consecutive nodes is set to 10 slots. Node 2 is upstream to node 3 and becomes active before node 3 does. Based on (3), we fix the value of CRM of node 2 and 3 to 156, 186 respectively. According to (1), these values of CRM should work out to guarantee the bandwidth requirements of the node. At the beginning of the simulation, the QA service node (node 1) takes the whole channel capacity, then node 2 becomes active after 400 slots time and node 3 becomes active 400 slots time later. As seen from Figure 2, before node 3 starts its transmission, node 2 is able to get 25% of the total bandwidth. As node 3 becomes active, the bandwidth of node 2 decreases to 20% instead of keeping to 25% as expected. Similar results are obtained even in the absence of QA service node (node 1). When switching the bandwidth requirement of node 2 and 3, the result shows that after node 3 becomes active the throughput of node 2 decreases to 37.5% from 40%. Several similar situations were detected during our simulations. This situation becomes worse as the number of active CO service nodes increases.

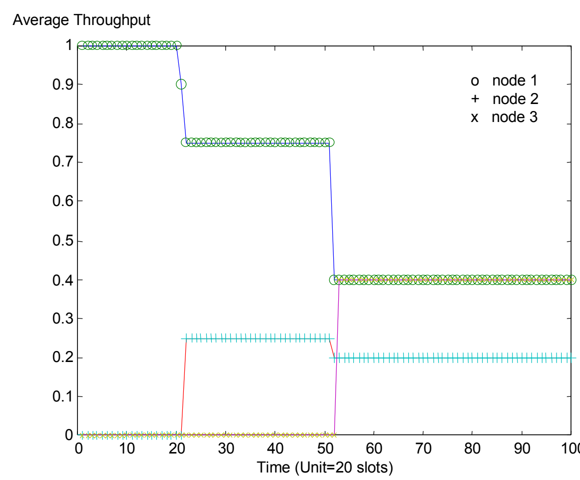

Fig. 2 Fixed CRM - node 2: 25%, node 3: 40%

In order to illustrate the source of the problem, we will use an example. According to [3], a node making use of the CO service requires placing a request for every segment to transmit. Under some traffic conditions, a node may have to delay its request due to the fact that other nodes may have already set the request bit of the passing slot. Consider the following scenario (depicted in Figure 3), which is based on the above example. Assume a sequence of slots and that node 3 has already set requests in slots 1, 4, 6, 9 and 11 according to CO service for its 40% of total bandwidth (we can get these index numbers through analyzing the CC with INC and SGC). Node 2 plans to set slots 1, 5 and 9 to request 25% of the network bandwidth. However, since node 3 has used slot 1, node 2 has to delay its request by one slot to set request in slot 2. As CRM limits the credit that node can accumulate, node 2 also has to delay its following request by one slot to slot 6, 10, and 14. Again, as slot 6 has been set by node 3, node 2 has to delay one slot to set slot 7, 11 and 15. Node 2 will find again that node 3 has also set slot 11 and its request will be delayed once more. Finally, node 2 can only set slots 2, 7, 12, 17 and so on through this procedure. Under this situation, the actual bandwidth node 2 can get is 20% of the total bandwidth (one slot in every five slots).

As the downstream CO service nodes have advantage to set requests on reverse bus, the upstream CO service nodes may experience problems in placing their requests. In order to overcome this problem, the upstream CO nodes should be allowed to transmit more often. In order to be able to do so, two possible methods can be considered: 1) to increase INC or 2) to change *CRM*.

The purpose to change *CRM* is to let a node to accumulate more credits. In this way, the node is allowed to access more often the network without changing its bandwidth requirement. If we fix the *CRM* according to (1), a node can set a request only at particular time slots. If there is a delay due to conflict with downstream node's requests, the node has to delay all the requests to follow. Thus, by increasing *CRM* dynamically when necessary, the node will not delay all following requests when forced to delay one request.

The purpose of increasing *INC* is to raise the bandwidth available to the CO service node. So changing *CRM* seems more feasible since it does not change the total CO service requirements of the network.

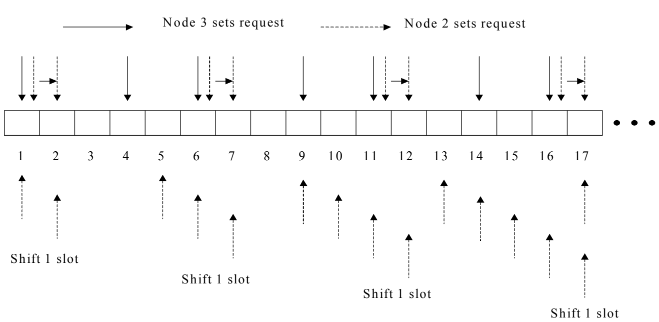

Fig. 3 Bandwidth depression of upstream CO service node.

The value of *CRM* should be changed dynamically allowing the node to adapt to the loading conditions of the network. In other words, *CRM* has to be changed to allow the node to compensate for the delays encountered, but without affecting the performance of other users. The main objective is to be able to accommodate the requirements of all active nodes as much as possible. (3) establishes the baseline on how to setup *CRM*. However in order to be able to compensate for delays encountered when multiple nodes attempt to access the channel, a node should be able to accumulate more credits while waiting to gain access to the network. For the case of two consecutive transmissions, this principle can be simply stated as follows:

$$CRM - SGC \geq SGC - (\lceil SGC/INC \rceil - i) \bullet INC \tag{5}$$

with $i = 1, 2, 3, \ldots$ and $i \leq \lceil SGC/INC \rceil$

which can be simply interpreted as follows. After a node has been successfully in sending a request and having paid SGC units to do so, it should be able to compensate for the delay by being allowed to transmit *i* slots before than normally expected.

The above equation defines the maximum value of the credit counter, *CRM*, as follows:

$$CRM \geq 2 \bullet SGC - (\lceil SGC/INC \rceil - i) \bullet INC \tag{6}$$

with $i = 1, 2, 3, \ldots$ and $i \leq \lceil SGC/INC \rceil$

This principle will be used to illustrate the method to dynamically adjust the value of CRM.

Another important issue is how to be able to detect if the node has got enough bandwidth to satisfy its requirement. This can be simply achieved by monitoring the number of segments transmitted during a period of *ls* slots. Furthermore, a margin ε of error can be defined to determine the level of accuracy. For a given bandwidth requirement, if the difference between the actual throughput and the node's requirement is greater than ε, the node is said to experience problems in getting its requirement fulfilled and its CRM should be increased. Otherwise, the CRM can be decreased or nothing will be changed depending on the value of *CRM*. The period of measurement depends on the value of ε. The period should be long enough so that the accuracy of measured throughput is smaller than ε. For generality, we should give a procedure taking into account different bandwidth requirements on how to change *CRM* compatible with the GBW protocol. For a detailed analysis of the general conditions, see appendix A.

1. Set $i = 0$ and $j = 1$;
2. Set $CRM = \lceil SGC/INC \rceil \bullet INC$;
3. Count the number of segments sent during *ls* slots;

4. IF | measured throughput – required capacity | < ε THEN goto 3;
5. IF measured throughput < required throughput THEN

    IF $i < \lceil SGC/INC \rceil$ THEN
        $i = i + 1$;
    ELSE
        $i = 0$ and $j = j + 1$;
  ELSE
    IF $i > 0$ THEN
        $i = i - 1$;
    ELSE
        IF $j > 1$ THEN
            $j = j - 1$
6. Set $CRM = j \bullet SGC - (\lceil SGC/INC \rceil - i) \bullet INC$;
7. Goto 3

We illustrate the proposed solution using a simple example. If a CO node requires 25% of bandwidth, it will set one request in every four slots. However, the node may be required to delay its requests when some other CO nodes downstream have already set the request bits of the corresponding slots. To illustrate this situation, we consider the following scenario.

As a CO node with a 25% of bandwidth requirement must wait for four slots passing by on reverse bus before it is allowed to send a request, the delay of a request results on the delay of the requests to follow. If we increase *CRM* of this node to another value, that can let the node send the following request only after three slots passing on the reverse bus, the node will be able to compensate the last delay. Thus, if the new *CRM* is increased to $2 \bullet SGC - 3 \bullet INC$ ($=5 \bullet INC$ because originally $SGC = 4 \bullet INC$ in this example), after sending a request the *CC* will decrease to $CRM - SGC = 5 \bullet INC - 4 \bullet INC = INC$. This idea is based on the following inequality:

$$CRM - SGC + (SGC/INC - 1) \bullet INC = CRM - SGC + 3 \bullet INC \geq SGC \quad (7)$$

After rearrangement, (7) becomes

$$CRM \geq 2 \bullet SGC - 3 \bullet INC = 5 \bullet INC \quad (8)$$

Then the CO node can send next request only after three slots passing by instead of four. The delay incurred in the last request is compensated by this action.

Figure 4 depicts the result of increasing *CRM* using the above method. Under the same situation as figure 2, increasing the *CRM* to $5 \bullet INC$ can properly solve the problem in allowing to acquiring 25% of the bandwidth. Any value of *CRM* below this one will not make any improvement.

6. Conclusion

We have derived a procedure to dynamically setup the system parameters of the CO data service. By using this mechanism, the CO data service is able to guarantee the throughput requirements of the application under various network conditions. Our results have shown the effectiveness of the proposed adaptive mechanism. Our future plans include a study of the CO service in the presence of bandwidth balancing scheme of the QA service as defined in [1].

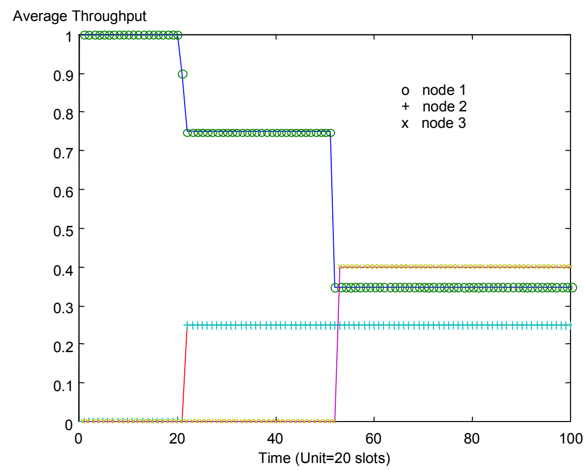

Fig. 4 Dynamic CRM - node 2: 25%, node 3: 40%

References:

[1] IEEE Standard 802.6-1990, *IEEE Standard for Local and Metropolitan Area Networks: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)*, July 1991.

[2] M.N.O. Sadiku, A.S. Arvind, Annotated Bibliography on Distributed Queue Dual Bus (DQDB), Computer Communications Review, 24, 1994, 21-36.

[3] IEEE Standard 802.6j, *Connection-Oriented Service on a Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Networks (MAN),* March 1995.

[4] D. Karvelas, *The Guaranteed Bandwidth and Balancing Mechanism for High Speed Metropolitan Area Networks*, ICC'94, New Orleans, 1994, 696-700.

[5] P. Martini, *Connection Oriented Data Service in DQDB*, Computer Networks and ISDN Systems, 26, 1994, 679-694.

Appendix:

In this appendix, we define the rules to change the value of *CRM* according to the changing conditions of the network and requirements of the applications. Initially, *CRM* is set using (3). According to (5), the new value of *CRM* is set as:

$$CRM \geq 2 \bullet SGC - (\lceil SGC/INC \rceil - 1) \bullet INC \quad (9)$$

while the *CRM* as specified by (3) is

$$CRM \geq \lceil SGC/INC \rceil \bullet INC \quad (10)$$

The following inequality should then be satisfied:

$$2 \bullet SGC - (\lceil SGC/INC \rceil - 1) \bullet INC \geq \lceil SGC/INC \rceil \bullet INC \quad (11)$$

Thus

$$2 \bullet SGC \geq 2 \bullet \lceil SGC/INC \rceil \bullet INC - INC \quad (12)$$

If we apply the equality, we obtain

$$2 \bullet SGC = 2 \bullet \lceil SGC/INC \rceil \bullet INC - INC \quad (13)$$

Because $SGC \geq INC$, the above equation can be written as

$$2 \bullet SGC = (2 \bullet n - 1) \bullet \mathrm{INC}, \quad n = 1, 2, 3, 4, \ldots \quad (14)$$

For n = 1($SGC = INC$), the inequality (11) is always true. For $n = 2$ ($2 \bullet INC \geq SGC > INC$), from (14) we obtain

$$SGC/INC = 3/2 \quad (15)$$

Thus, if $2 \bullet SGC/3 > INC \geq SGC/2$, i.e., $\frac{SGC}{INC} \in \left(\frac{3}{2}, 2\right]$

the inequality (11) holds true. Otherwise, if ($SGC > INC \geq 2 \bullet SGC/3$), the new *CRM* will not be greater than the old one. In this case ,*CRM* should be changed as follows:

$$CRM - SGC + (\lceil SGC/INC \rceil - 2) \bullet INC \geq SGC \quad (16)$$

Since $\lceil SGC/INC \rceil = 2$ under this condition, the inequality can be written as

$$CRM \geq 2 \bullet SGC \quad (17)$$

The new value of CRM will be $2 \bullet SGC$.
When n = 3 ($3 \bullet INC \geq SGC > 2 \bullet INC$), equation (13) becomes

$$SGC/INC = 5/2 \quad (18)$$

Then if $2 \bullet SGC/5 > INC \geq SGC/3$, i.e.,

$$\frac{SGC}{INC} \in \left(\frac{5}{2}, 3\right] \quad (19)$$

the inequality (11) is true. Otherwise, we apply the same strategy. Then the CRM follows

$$\mathrm{CRM} - \mathrm{SGC} + (\lceil \mathrm{SGC/INC} \rceil - 2) \bullet \mathrm{INC} \geq \mathrm{SGC} \quad (20)$$

After rearrangement,(20) and under (19) becomes

$$CRM \geq 2 \bullet SGC - INC \quad (21)$$

When $n = 4$ ($4 \bullet INC \geq SGC > 3 \bullet INC$),
if $2 \bullet SGC/7 > INC \geq SGC/4$, i.e.,

$$\frac{SGC}{INC} \in \left(\frac{7}{2}, 4\right] \quad (22)$$

the new CRM comes from (11). Else we apply (16).
When n = 5 ($5 \bullet INC \geq SGC > 4 \bullet INC$), if $2 \bullet SGC/9 > INC \geq SGC/5$, i.e.,

$$\frac{SGC}{INC} \in \left(\frac{9}{2}, 5\right] \quad (23)$$

the new *CRM* comes from (11). Else we apply (16). In the case that *CRM* should be increased more than once, the procedure works as follows:

First, CRM is changed using the following inequality:

$$CRM \geq \mathrm{j} \bullet SGC - (\lceil SGC/INC \rceil - i) \bullet INC \quad (24)$$

where j and i are integers, $j \geq 2$ and i is from 0 to $\lceil SGC/INC \rceil$. The initial value of j is 2 and the initial value of i is 0. When *CRM* is increased by the first time, i is set to 1 or 2 based on the value of SGC/INC. The next step consists in increasing i by 1 to get a new *CRM* greater than the current one being used. When i reaches $\lceil SGC/INC \rceil$, we reset i to 0 and increase j by 1. When decreasing *CRM*, i is first decreased. When i reaches 0, it is reset to $\lceil SGC/INC \rceil$ and j is decreased by 1 until *CRM* reaches its initial value, given by (1).