TR 101 123 V1.1.1 (1997-11)

Technical Report

Broadband Integrated Services Digital Network (B-ISDN); Resource management procedures and cases of their possible usage

European Telecommunications Standards Institute

Reference

DTR/NA-052809 (aec00ics.PDF)

Keywords

ATM, B-ISDN, control, management, traffic

ETSI Secretariat

Postal address

F-06921 Sophia Antipolis Cedex - FRANCE

Office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

X.400

c= fr; a=atlas; p=etsi; s=secretariat

Internet

secretariat@etsi.fr
http://www.etsi.fr

Contents

Intellectual Property Rights

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Network Aspects (NA).

1 Scope

The present document is to define a specific resource management protocol to support the transfer ABT-DT capability and to consider some specific cases of its possible application. The material contained in the present document is strictly informative to give guidance for implementation in networks. No additional ATM transfer capabilities as compared to ITU-T Recommendation I.371 [1] are introduced.

The present document specifies the functions supported by the ABT-DT protocol, the protocol messages and the protocol procedures. In order to achieve an effective resource management capability, these functions shall be complemented by additional functions (e.g. connection policing, resource allocation) that are network operator specific and are not covered by the present document.

2 References

References may be made to:

a) specific versions of publications (identified by date of publication, edition number, version number, etc.), in which case, subsequent revisions to the referenced document do not apply; or

b) all versions up to and including the identified version (identified by "up to and including" before the version identity); or

c) all versions subsequent to and including the identified version (identified by "onwards" following the version identity); or

d) publications without mention of a specific version, in which case the latest version applies.

A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] ITU-T Recommendation I.371: "Traffic Control and Congestion Control in B-ISDN";

3 Abbreviations

For the purposes of the present document the following abbreviations apply:

ABT	ATM Block Transfer
ABT-DT	ATM Block Transfer with Delayed Transmission
ATM	Asynchronous Transfer Mode
BCR	Block Cell Rate
B-VPN	Broadband Virtual Private Network
FRM	Fast Resource Management
IBT	Intrinsic Burst Tolerance
OAM	Operation And Maintenance
QoS	Quality of Service
RM	Resource Management
SCR	Sustainable Cell Rate

4 General principles

The protocol specified in the present document is an application of Fast Resource Management (FRM) techniques, specifically designed to support the ABT/DT transfer capability.

FRM techniques operate on the time scale of the round-trip propagation delay of an ATM connection and allow the modification of the ATM resources allocated to a connection depending on the actual need of the connection and/or on the network status. A basic characteristic of FRM procedure is that the command messages required by the specific protocol are transported "in band" into the data flow by means of specific Resource Management (RM) cells. This is intended to improve the protocol response time.

The general format of RM cells is specified in ITU-T Recommendation I.371 [1] as well as the specific RM cell format used by the ABT/DT protocol.

5 ABT/DT Protocol description

5.1 Introduction

This subclause presents the specification of a protocol for supporting the ATM Block Transfer with Delayed Transmission (ABT/DT) capability. The high level description of ABT/DT as well as the definition of the ATM block are given in ITU-T Recommendation I.371 [1]. The basic principle behind ABT/DT is that the information is transmitted in block of cells (ATM Blocks) and that the bandwidth available for transmission, i.e., the Block Cell Rate (BCR), is negotiated individually for each ATM Block. ATM Blocks are delineated by specific messages of the ABT/DT protocol, as will be specified later on in the present document.

In the present specification, only point-to-point bi-directional communications are considered and the protocol is limited to a single network; point-to-multipoint communications are for further study.

Even though the ABT/DT transfer capability may usually involve bi-directional data transmission, for the sake of simplicity the description provided by the present document refers only to a single direction of data transmission, the generalization to the bi-directional case being straightforward. With this restriction the reference configuration for an ABT/DT connection is shown in figure 1. The terminal originating the data is said to be the *Source* terminal and the terminal receiving the data is said to be the *Destination* terminal. Between the two terminals there is a mono-directional data flow (from the source toward the destination) and a bi-directional control flow realized by means of RM cells. The direction of the control flow going from the source toward the destination is said to be the *Forward direction*; the reverse direction is said to be the *Backward direction*. Furthermore, the network node closest to the source is said to be the *Ingress node*; the one closest to the destination is said to be the *Egress node*.

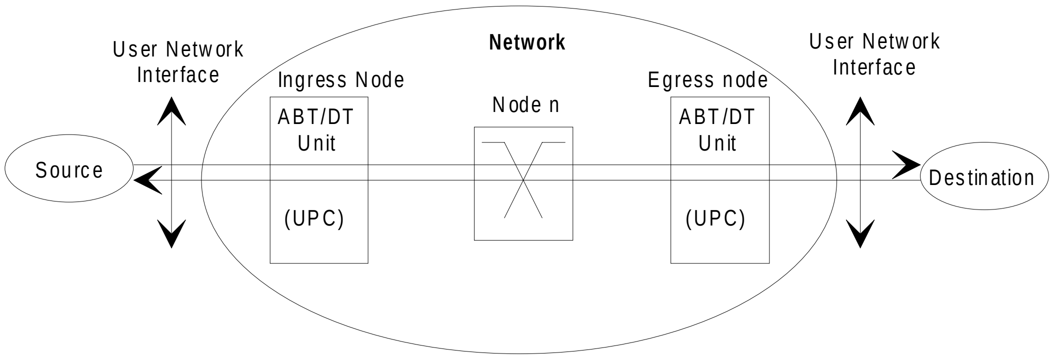

Figure 1: Reference configuration for ABT/DT

Each direction of transmission is realized by means of an ABT/DT connection, whose traffic contract specifies:

1) -The peak cell rate value for each cell flow of the connection, namely:

   - the maximum cell rate $\Lambda_{max(0+1)}$ of the user generated CLP=0+1 cell flow (including user OAM but no RM cells);

- the maximum cell rate $\Lambda_{max(OAM)}$ of the user OAM cell flow. Note that in the signalling message this peak cell rate is declared as a proportion of the aggregate peak cell rate $\Lambda_{max(0+1)}$.

Only the first parameter is mandatory; the second one is optional. Values of these two parameters are upperbounds for the respective negotiable BCR values of the different cell flows.

2) The CDV tolerance associated with each maximum cell rate, namely:

- $\tau_{max(0+1)}$ of the CLP=0+1 cell flow (mandatory);

- $\tau_{max(OAM)}$ of the OAM cell flow (optional and possibly implicit).

CDV Tolerances may be implicitly or explicitly declared (note 1).

3) The maximum frequency of BCR negotiations by means of:

- the peak cell rate $1/T_{RM}$ of the user request RM cells.

4) The CDV tolerance τ_{RM} associated with the peak cell rate of the user request RM cell flow.

5) The committed bandwidth of the CLP=0+1 cell flow by means of a Sustainable Cell Rate (SCR) and an Intrinsic Burst Tolerance (IBT); the SCR may be set equal to 0.

6) The requested QoS class.

NOTE 1: The CDV tolerance τ associated with the CLP=0+1 cell flow may depend on the peak cell rate λ of this cell flow, namely τ is a function $\tau(\lambda)$ of λ. In that case, the function $\tau(\lambda)$ is specified on a subscription basis and $\tau_{max}(0+1)=\tau(\Lambda_{max}(0+1))$. The CDV tolerance associated with OAM traffic should satisfy the general requirements given in Appendix II to ITU-T Recommendation I.371 [1].

An ABT/DT connection is subject to Connection Admission Control (CAC) on the basis of the declared parameters (peak and sustainable cell rates). At the establishment of an ABT/DT connection, the connectivity between the source and the destination is assured but no resources are allocated; the connection is then said *dormant*. Resources are allocated only via BCR negotiation by means of RM cells and the connection is then said *active*.

5.2 Notation and conventions

The following conventions are used in the present document in order to identify the different cell flows generated by a user terminal:

- OAM cell flow: includes only end to end OAM cells generated by the user;

- RM cell flow: includes any kind of RM cells generated by the user;

- user data CLP=0+1 cell flow: includes all user data cells, regardless the value of the CLP bit;

- aggregated CLP=0+1 cell flow: includes all user data cells and OAM cells, but not RM cells.

Furthermore, the following symbols will be used in the rest of the present document:

- $\Lambda(0+1)$: a generic value of the BCR for the aggregated CLP=0+1 cell flow;

- $\Lambda(OAM)$: a generic value of the BCR for the OAM cell flow;

- $\Lambda_{current}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow currently allocated to a connection at the Ingress node;

- $\Lambda_{Current}(OAM)$: value of the BCR for the OAM cell flow currently allocated to a connection at the Ingress node;

- $\lambda_{Current}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow currently allocated to a connection at the Egress node;

- $\lambda_{Current}(OAM)$: value of the BCR for the OAM cell flow currently allocated to a connection at the Egress node;

- $\Lambda_N(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow currently allocated to a connection at the generic node N;
- $\Lambda_N(OAM)$: value of the BCR for the OAM cell flow currently allocated to a connection at the generic node N;
- $\Lambda'_{Current}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow currently reserved but not yet allocated to a connection at the Ingress node;
- $\Lambda'_{Current}(OAM)$: value of the BCR for the OAM cell flow currently reserved but not yet allocated to a connection at the Ingress node;
- $\lambda'_{Current}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow currently reserved but not yet allocated to a connection at the Egress node;
- $\lambda'_{Current}(OAM)$: value of the BCR for the OAM cell flow currently reserved but not yet allocated to a connection at the Egress node;
- $\Lambda_{Req}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow requested by the source;
- $\Lambda_{Req}(OAM)$: value of the BCR for the OAM cell flow requested by the source;
- $\lambda_{Req}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow requested by the destination;
- $\lambda_{Req}(OAM)$: value of the BCR for the OAM cell flow requested by the destination;
- $\Lambda_{alloc}(0+1)$: value of the BCR for the aggregated CLP=0+1 cell flow that has been allocated following a negotiation request;
- $\Lambda_{Alloc}(OAM)$: value of the BCR for the OAM cell flow that has been allocated following a negotiation request;

5.3 Functions supported by the ABT/DT protocol

5.3.1 BCR negotiation functions

The BCR negotiation functions are the basic functions supported by the ABT-DT protocol. By means of these functions the user terminals may negotiate with the network new values for the BCR of both the aggregated CLP=0+1 cell flow ($\Lambda(0+1)$) and the OAM cell flow ($\Lambda(OAM)$). The negotiation may be intended at achieving either a BCR decrease or a BCR increase.

The BCR negotiation can be initiated either by the source terminal (*forward BCR negotiation*) or by the destination terminal (*backward BCR negotiation*). In the case of concurrent requests, one initiated by the source and the other initiated by the destination, the latter takes the precedence.

Finally, the BCR negotiations can be carried out in two different modes: the *Rigid mode* and the *Elastic mode*.

The rigid mode is characterized by the fact that the BCR value specified by the user in its request can not be modified (i.e., lowered) by the network, so that the request is accepted only if all the network elements along the path of the connection are able to allocate the requested amount of resources.

The elastic mode is characterized by the fact that the BCR value specified by the user in its request can be modified (i.e., lowered) by the network, so that a request can be accepted also if some of the network elements along the path of the connection are able to allocate only an amount of resources lower than the one requested by the user.

By convention, BCR negotiations intended at achieving a BCR decrease are carried out only in the rigid mode; therefore, there are compulsively six types of BCR negotiation functions:

1) **Rigid Forward BCR decrease**: allow the source to negotiate a lower BCR for the aggregated CLP=0+1 cell flow and/or for the OAM cell flow;

2) **Rigid Forward BCR increase**: allow the source to negotiate a higher BCR for the aggregated CLP=0+1 cell flow and/or for the OAM cell flow;

3) **Rigid Backward BCR decrease**: allow the destination to negotiate a lower BCR for the aggregated CLP=0+1 cell flow and/or for the OAM cell flow.

4) **Rigid Backward BCR increase**: allow the destination to negotiate a higher BCR for the aggregated CLP=0+1 cell flow and/or for the OAM cell flow.

5) **Elastic Forward BCR increase**: allow the source to negotiate a higher BCR for the aggregated CLP=0+1 cell flow and/or for the OAM cell flow;

6) **Elastic Backward BCR increase**: allow the destination to negotiate a higher BCR for the aggregated CLP=0+1 cell flow and/or for the OAM cell flow.

The Rigid Backward BCR decrease function and the Rigid Backward BCR increase function are realized through the same protocol procedure (see subclause 5.6.1.3). A BCR decrease is always in the rigid mode.

5.3.2 User maintenance functions

User maintenance functions allow a user terminal to know the BCR value currently allocated within the network and to make it equal all along the connection. These functions can be used if some discrepancy is observed between the BCR values known at the source and at the destination and the BCR value allocated within the network. The use of these functions can not implicate the allocation of new resources to a connection, but only the alignment of the amount of allocated resources to the minimum among the values currently allocated by the different network elements along the path of the connection.

User maintenance functions can be initiated either by the source or by the destination and are always carried out in the elastic mode. Therefore, there are two types of user maintenance functions:

1) **Forward Status Enquiry**: is a user maintenance function initiated by the source;

2) **Backward Status Enquiry**: is a user maintenance function initiated by the destination;

5.3.3 Traffic control functions

Traffic Control functions are initiated by the network under certain conditions (e.g., block non conformance, presence of congestion etc.) to force a modification (usually a reduction) of the BCR allocated to a connection. These functions are similar to BCR negotiation functions, apart from the fact that they are initiated by a network node instead of a user terminal. They can be initiated either by the ingress node (*Forward Traffic Control*) or by the egress node (*Backward Traffic* Control) and can be carried out either in the *Rigid mode* or in the *Elastic mode*. Traffic Control functions have priority over BCR Negotiation functions initiated by either user.

Specifically, if a BCR Negotiation is in progress while a traffic control procedure is running, then some of the messages corresponding to the BCR Negotiation may be discarded at the Ingress node or at the Egress node. Moreover, upon reception of Traffic Control message at a network node, any BCR reservation performed by either user is cancelled. If a BCR negotiation is initiated by the source (respectively the destination) while a Traffic Control message issued at network ingress (respectively egress), the source (respectively the destination) should receive a Not_Ready message.

Finally, Traffic Control messages generated at the Ingress node have priority over those generated at the Egress node. As above, some messages relative to the traffic control procedure at network egress may be discarded.

Rigid Traffic Control functions are typically used by the network to verify if the BCR currently allocated to a connection is still available and, if this is not the case, to change it at a lower value.

Elastic Traffic Control functions are used by the network to perform an elastic re-negotiation of the BCR allocated to a connection and to notify the new allocated value to the source. The new value may either be lower or higher than the previous one.

Summarizing, there are four types of Traffic Control Functions:

1) **Rigid Forward Traffic Control**: is a Rigid Traffic Control function initiated by the ingress node;

2) **Rigid Backward Traffic Control**: is a Rigid Traffic Control function initiated by the egress node;

3) **Elastic Forward Traffic Control**: is an Elastic Traffic Control function initiated by the ingress node;

4) **Elastic Backward Traffic Control**: is an Elastic Traffic Control function initiated by the egress node.

5.3.4 Network maintenance functions

Network Maintenance functions are used by the network to force to some specific value the BCR allocated to a connection in each network node along the path of the connection or to retry a transaction. These functions are typically used to recover from anomalous conditions and are triggered by the expiration of some of the internal timers of the protocol. These are for instance Time-Out Handling functions.

5.3.4.1 Time-out handling functions

Time-Out Handling functions are Network Maintenance functions triggered by the expiration of internal timers of the protocol due, for instance, to the loss of some messages during a BCR negotiation procedure or a Traffic Control procedure. This ETR defines two Time-Out Handling functions:

1) **Recovering from Ingress Time-Out**: this function is triggered by the expiration of W_Ingress timer at the network ingress (see subclause 5.6.4.1.1);

2) **Recovering from Egress Time-Out**: this function is triggered by the expiration of W_Egress timer at the network egress (see subclause 5.6.4.1.1).

Additional Time-Out Handling functions are for further studies.

5.3.4.2 Other Network Maintenance functions

Additional Network Maintenance functions are for further studies.

5.4 RM cell format

The RM cell format used for ABT/DT is specified in ITU-T Recommendation I.371 [1]. Within the present document the different fields of the RM cells are referred to according to the following nomenclature:

- M.Type: indicates the Req/Ack bit of the Message Type field and discriminates between Request cells (M.Type=0) and Acknowledgement cells (M.Type=1);

- M.Dir: indicates the Direction bit of the Message Type field and discriminates between cells flowing in the forward direction (M.Dir=0) and cells flowing in the backward direction (M.Dir=1);

- M.Elastic: indicates the Elastic/Rigid bit of the Message Type field and discriminates between the elastic mode (M.Elastic=0) and the Rigid mode (M.Elastic=1);

- M.CI: indicates the Congestion Indication bit of the Message Type field and indicates whether the network is congested (M.CI=1) or not (M.CI=0); note that, in the elastic mode, this field can be used to indicate whether the connection has received at least the bandwidth corresponding to its fair share (M.CI=0) or not (M.CI=1);

- M.Maintenance: indicates the Maintenance bit of the Message Type field and discriminates between RM cells devoted to user or network maintenance functions (M. Maintenance=1) and RM cells devoted to BCR negotiation and traffic control functions (M.Maintenance=0);

- M.Trf: indicates the Traffic Management bit of the Message Type header and discriminates between RM cells devoted to Network Traffic Management functions (M.Trf=1) and RM cells devoted to BCR negotiation or user maintenance functions (M.Trf=0);

- M.Rate(0+1): indicates the CLP=0+1 BCR field;

- M.Rate(OAM): indicates the User OAM BCR field;

- M.Number: indicate the Sequence Number field; when used, this field is incremented by one (modulo 2^{32}) in each subsequent RM cell transmitted by a given entity. On the contrary, when not used, it is set equal to 0;

- M.Size: indicate the Block Size field; this field is not used in this version of the protocol.

5.5 Protocol messages

This subclause defines the protocol messages used for implementing the functions described in subclause 5.3. Each message is identified by the values of the Message Type binary fields in the RM cells.

Messages are classified into three main categories according to their utilization:

- BCR Negotiation messages;
- User Maintenance messages;
- Network Traffic Management messages,

and are listed in the form of tables that specify the coding of the Message Type binary fields used by each message. The characters "--" mean that either the value 0 or the value 1 can be used for the corresponding field and that the value can be changed within the network.

5.5.1 BCR negotiation messages

These messages are identified by M.Maintenance=0 and M.Trf=0. The M.Elastic field discriminates between messages related to elastic negotiation functions (M.Elastic=0) and messages related to rigid negotiation functions (M.Elastic=1). The M.Dir field discriminates between messages flowing in the forward direction (M.Dir=0) and messages flowing in the backward direction (M.Dir=1). The M.Type field discriminates between request messages (M.Type=0) and Acknowledgement messages (M.Type=1). The M.CI field does not identifies any specific message, but it is used to indicate whether congestion has been encountered in the network (M.CI=1) or not.

5.5.1.1 Rigid BCR negotiation

The four messages listed in table 1 are devoted to support the Rigid BCR Negotiation functions defined in subclause 5.3.1. Two of them are Bandwidth Request messages (M.Type=0), whereas the remaining two messages are Bandwidth Acknowledgement messages (M.Type=1).

Table 1: Messages for Rigid BCR Negotiation

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
R_Forward_User_Req	0	0	1	--	0	0
R_Backward_User_Req	0	1	1	--	0	0
R_Backward_Net_Ack	1	1	1	--	0	0
R_Forward_User_Ack	1	0	1	--	0	0

5.5.1.2 Elastic BCR negotiation

The four messages listed in table 2 are devoted to support the Elastic BCR Negotiation functions defined in subclause 5.3.1.2. Two of them are Bandwidth Request messages (M.Type=0), whereas the remaining two messages are Bandwidth Acknowledgement messages (M.Type=1).

Table 2: Messages for Elastic BCR Negotiation

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
E_Forward_User_Req	0	0	0	--	0	0
E_Backward_User_Req	0	1	0	--	0	0
E_Backward_Net_Ack	1	1	0	--	0	0
E_Forward_User_Ack	1	0	0	--	0	0

5.5.2 User maintenance messages

These messages are identified by M.Maintenance=1 and M.Trf=0. The M.Elastic field is always set to 0. The M.Dir field discriminates between messages flowing in the forward direction (M.Dir=0) and messages flowing in the backward direction (M.Dir=1). The M.Type field discriminates between request messages (M.Type=0) and Acknowledgement or Not Ready messages (M.Type=1). The M.CI field is always set to 0.

5.5.2.1 Forward and backward status enquiry

The four messages listed in table 3 are devoted to support the Forward and Backward Status Enquiry functions defined in subclause 5.3.2. Two of them are Request messages (M.Type=0), whereas the remaining two messages are Acknowledgement messages (M.Type=1).

Table 3: Messages for forward and backward status enquiry functions

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
M_Forward_User_Req	0	0	0	0	1	0
M_Backward_User_Req	0	1	0	0	1	0
M_Backward_Net_Ack	1	1	0	0	1	0
M_Forward_User_Ack	1	0	0	0	1	0

5.5.2.2 Not ready

The two messages listed in table 4 are the Not Ready messages generated by the network ingress or the network egress nodes when they are unable to accept the BCR negotiation request forwarded by the source or by the destination. The M.Elastic field should be set equal to 0.

Table 4: Not ready messages

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
M_Forward_Notready_Ack	1	0	0	1	1	0
M_Backward_Notready_Ack	1	1	0	1	1	0

5.5.3 Traffic control messages

These messages are identified by M.Maintenance=0 and M.Trf=1. The M.Elastic field discriminates between messages related to the Elastic mode (M.Elastic=0) end messages related to the rigid mode (M.Elastic=1). The M.Dir field discriminates between messages flowing in the forward direction (M.Dir=0) and messages flowing in the backward direction (M.Dir=1). The M.Type field discriminates between request messages (M.Type=0) and Acknowledgement or Not Ready messages (M.Type=1). The M.CI field does not identify any specific message, but it is used to indicate whether congestion has been encountered in the network (M.CI=1) or not (M.CI=0).

5.5.3.1 Rigid traffic control

The four messages listed in table 5 are devoted to support the Rigid Traffic Control functions defined in subclause 5.3.3. Two of them are Request messages (M.Type=0), whereas the remaining two messages are Acknowledgement messages (M.Type=1).

Table 5: Messages for rigid traffic control functions

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
R_Traffic_Forward_Req	0	0	1	--	0	1
R_Traffic_Backward_Req	0	1	1	--	0	1
R_Traffic_Backward_Ack	1	1	1	--	0	1
R_Traffic_Forward_Ack	1	0	1	--	0	1

5.5.3.2 Elastic traffic control

The four messages listed in table 6 are devoted to support the Elastic Traffic Control functions defined in subclause 5.3.3. Two of them are Request messages (M.Type=0), whereas the remaining two messages are Acknowledgement messages (M.Type=1).

Table 6: Messages for elastic traffic control functions

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
E_Traffic_Forward_Req	0	0	0	--	0	1
E_Traffic_Backward_Req	0	1	0	--	0	1
E_Traffic_Backward_Ack	1	1	0	--	0	1
E_Traffic_Forward_Ack	1	0	0	--	0	1

5.5.4 Network maintenance messages

These messages are identified by M.Maintenance=1 and M.Trf=1. The M.Elastic field is takes the value of the M.Elastic field of the transaction which is pending. The M.Dir field discriminates between messages flowing in the forward direction (M.Dir=0) and messages flowing in the backward direction (M.Dir=1). The M.Type field discriminates between request messages (M.Type=0) and Acknowledgement or Not Ready messages (M.Type=1). The M.CI field does not identifies any specific message, but it is used to indicate whether congestion has been encountered in the network (M.CI=1) or not.

5.5.4.1 Time-out handling

The four messages listed in table 7 are devoted to support the Time-Out Handling functions defined in subclause 5.3.4.1. Two of them are Request messages (M.Type=0), whereas the remaining two messages are Acknowledgement messages (M.Type=1).

Table 7: Messages for time-out handling functions

Primitive’s name	M.Type	M.Dir	M.Elastic	M.CI	M.Maint.	M.Trf
N_Forward_Req	0	0	--	--	1	1
N_Backward_Req	0	1	--	--	1	1
N_Backward_Ack	1	1	--	--	1	1
N_Forward_Ack	1	0	--	--	1	1

5.6 Normal protocol procedures

This subclause describes the protocol procedures executed under normal conditions, i.e., in the assumption that no collisions happen between different procedures. The problem of collisions will be dealt with in subclause 5.7.

5.6.1 BCR negotiation procedures

5.6.1.1 General principles

Any BCR negotiation (BCR increase or decrease) by either user of a bi-directional ABT/DT communication is achieved by sending a Bandwidth Request message into the network and by waiting for a Bandwidth Acknowledgement message from the network, except in the case of a BCR decrease initiated by the source itself, which is immediately taken into account by the network. If positive, this Bandwidth Acknowledgement message should be acknowledged by the source by sending another Bandwidth Acknowledgement message. The source can start the transmission of a new ATM block only after having sent this Bandwidth Acknowledgement message.

ATM blocks are delineated by one of the following messages:

- Bandwidth Request messages indicating a BCR decrease initiated by the source;
- Bandwidth Acknowledgement messages sent by the source.

A Bandwidth Request message when sent by a given entity (either the source or the destination) should convey the respective requested BCR values for the CLP=0+1 and OAM cell flows.

The BCR values conveyed by a Bandwidth Request message shall be lower than the maximum values negotiated at connection set up $\Lambda_{Max}(0+1)$ and $\Lambda_{Max}(OAM)$; therefore, the fields M.Rate(0+1) and M.Rate(OAM) of these messages are checked and enforced at the Ingress node (Forward Bandwidth Request messages) or at the Egress node (Backward Bandwidth Request messages) as follows:

$$\begin{cases} \text{M. Rate}(0+1) := \min\{\text{M. Rate}(0+1), \Lambda_{\max}(0+1)\}, \\ \text{M. Rate(OAM)} := \min\{\text{M. Rate(OAM)}, \Lambda_{\max}(\text{OAM}), \text{M. Rate}(0+1)\}. \end{cases} \qquad (1)$$

A Bandwidth Acknowledgement message or a Bandwidth Request message when received by the destination should convey the respective allocated BCR values $\Lambda_{Alloc}(0+1)$ and $\Lambda_{Alloc}(OAM)$ for the CLP=0+1 and OAM cell flows. The fields M.Rate(0+1) and M.Rate(OAM) of Bandwidth Acknowledgement messages sent by the source are checked at the ingress node and forced according to:

$$\begin{cases} \text{M. Rate}(0+1) := \min\{\Lambda'_{\text{Current}}\ (0+1), \text{M. Rate}(0+1)\}, \\ \text{M. Rate(OAM)} := \min\{\Lambda'_{\text{Current}}\ (\text{OAM}), \text{M. Rate(OAM)}, \text{M. Rate}(0+1)\}. \end{cases} \qquad (2)$$

Upon the reception of a Bandwidth Request message from the user (either the source or the destination) each network node establishes whether the request is a Bandwidth Increase Request or a Bandwidth Decrease Request by looking at the fields M.Rate(0+1) and M.Rate(OAM), if

$$\text{M.Rate}(0+1) \leq \Lambda_{Current}(0+1) \qquad (3)$$

then the request is a Bandwidth Decrease Request; otherwise it is a Bandwidth Increase Request.

By convention, Bandwidth Decrease Requests shall be forwarded in the rigid mode only; it is an implementation option how to deal with Bandwidth Decrease Requests forwarded in the elastic mode; for instance, these requests can be:

- discarded at the Ingress or Egress node;
- changed into rigid requests by forcing the M.Elastic field to 0 at the Ingress or Egress node.

Only one BCR negotiation initiated by a given entity (the source or the destination) may be in progress within the network in a given direction. If the same entity initiates a BCR negotiation in a given direction while another one is still in progress within the network, then the network discards the new Bandwidth Request message and sends a Not Ready message back to the originating entity.

5.6.1.2 Rigid BCR negotiation procedures

5.6.1.2.1 Rigid forward BCR decrease

A Rigid Forward BCR Decrease procedure is initiated by the source by sending a R_Forward_User_Req message (see table 1) which satisfies (3).

Immediately after issuing such a message, the source should adapt its transmission rate in order to conform to the new BCR values (and associated CDV tolerances), since the message delineates a new ATM block.

The flow of messages for the Rigid Forward BCR Increase procedure is shown in figure 3.

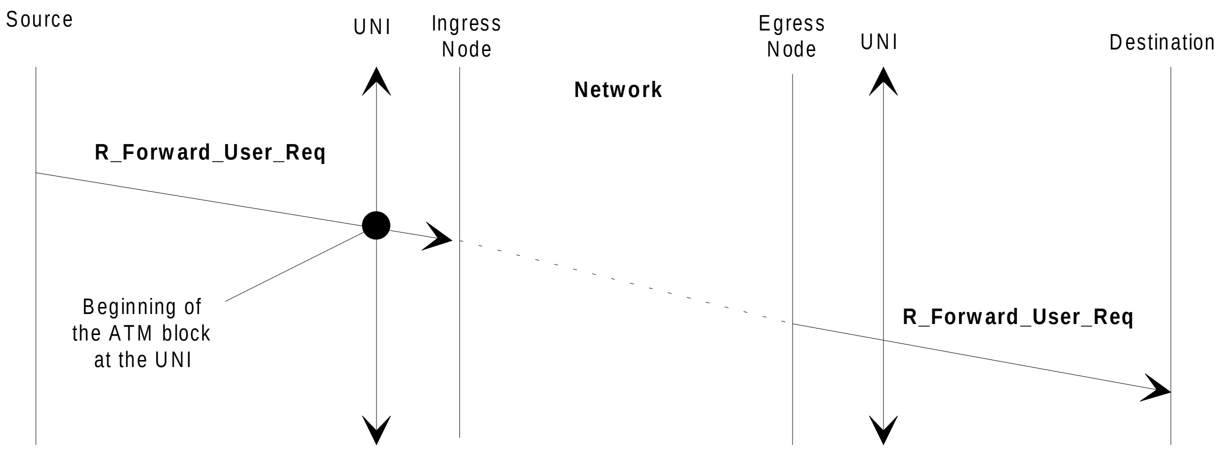

Figure 2: Rigid Forward BCR Decrease procedure with no collision

At the Ingress node, the requested BCR values are enforced as in (1) and then checked to see if inequality (3) is satisfied. If this is the case, then $\Lambda_{Current}(0+1)$ and $\Lambda_{Current}(OAM)$ are updated to the new BCR values and the message is forwarded to the network nodes. Each node receiving the message checks that inequality (3) is verified and immediately updates the bandwidth allocated to the connection. . A BCR decrease is not acknowledged by the network.

5.6.1.2.2 Rigid forward BCR increase

A Rigid Forward BCR Increase procedure is initiated by the source by sending a R_Forward_User_Req (see table 1) message with:

$$M.Rate(0+1) > \Lambda_{current}(0+1)$$

In this case, the request message does not delineate a new ATM block and the source shall wait for the acknowledgement from the network before updating its transmission rate.

The flow of messages for the Rigid Forward BCR Increase procedure is shown in figure 3.

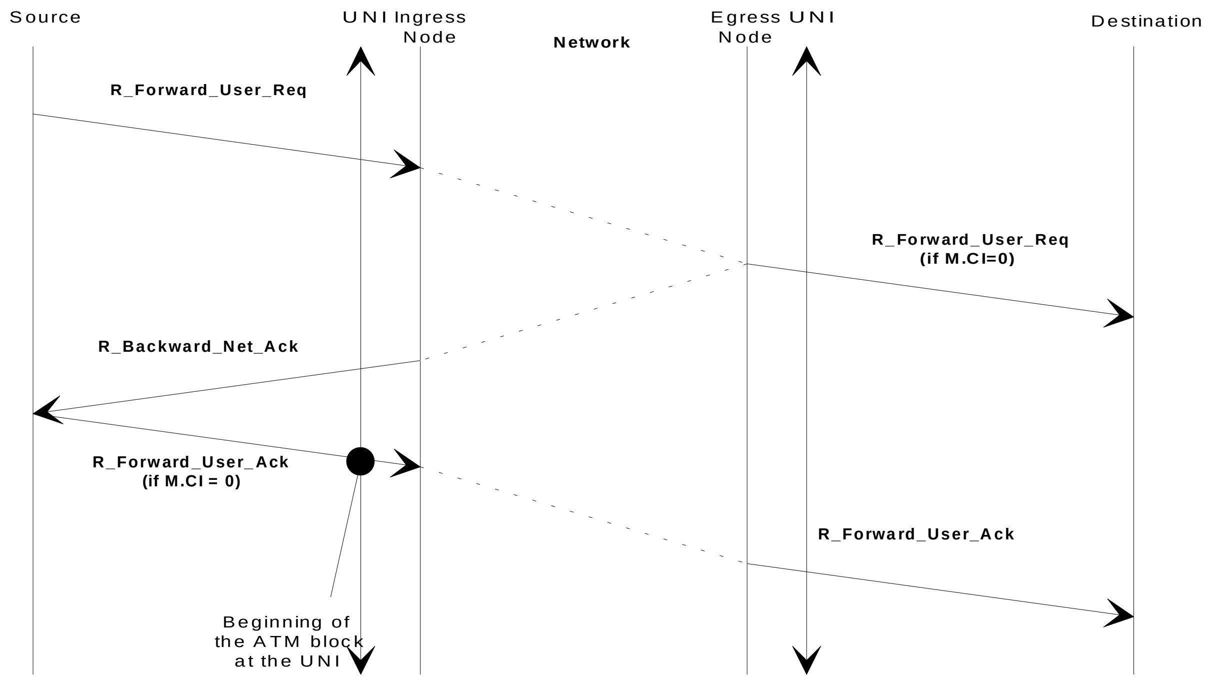

Figure 3: Rigid Forward BCR Increase procedure with no collision

At the Ingress node the requested BCR values are enforced according to (1) and checked to see whether inequality (3) is satisfied. Since the message initiates a BCR Increase procedure inequality (3) will not be satisfied. The Ingress node then processes the message and accepts or denies the bandwidth increase request according to a specific resource allocation policy. If the requested resources are available, then they are reserved (but not yet allocated); otherwise, the request is denied and the M.CI field is set equal to 1. In both cases the message is forwarded into the network. Each node receiving the message checks first the M.CI field: if the M.CI field is equal to 1 the message is forwarded to the next node and no other actions are taken (the bandwidth increase request has already been rejected). Otherwise the message is processed to check the availability of the requested resources as in the Ingress node and it is then forwarded to the next node.

A R_Forward_User_Req message reaching the Egress node with M.CI=1 is discarded after having generated the transmission of a R_Backward_Net_Ack message with M.CI=1 back to the source. The R_Backward_Net_Ack message travelling into the network cancels the bandwidth reservation in all the nodes that had been able to reserve the requested resources and it is then forwarded to the source that does not has any other action to take. The reception of this message terminates the BCR negotiation phase.

A R_Forward_User_Req message reaching the Egress node with M.CI=0 is processed as usual. If the requested resources are not available the message is discarded and a R_Backward_Net_Ack message with M.CI=1 is sent back to the source as in the previous case. Otherwise, the message is forwarded to the destination and a R_Backward_Net_Ack message with M.CI=0 is sent back to the source. This message carries into the M.Rate(0+1) and M.Rate(OAM) fields the BCR values that have been reserved for the connection. Each node receiving this message forwards it to the following node (in the backward direction) without the need to take any special action (it may optionally read the reserved BCR values and update its internal values).

Once the message is received by the source, it triggers the transmission of a R_Forward_User_Ack message with M.CI=0. This message should carry values lower than or equal to the reserved BCR and delineates a new ATM block. Immediately after its transmission the source is allowed to update its transmission rate.

At the Ingress node the M.Rate(0+1) and M.Rate(OAM) fields of the R_Forward_User_Ack message are enforced according to (2); then the reserved resources are actually allocated to the connection and the message is forwarded into the network up to the destination. Each node receiving the message allocates the reserved resources. The reception of this message terminates the BCR negotiation phase.

5.6.1.2.3 Rigid backward BCR increase or decrease

A Rigid Backward BCR Increase or Decrease procedure is initiated by the destination by sending a R_Backward_User_Req message.

The flow of messages for this procedure is shown in figure 4.

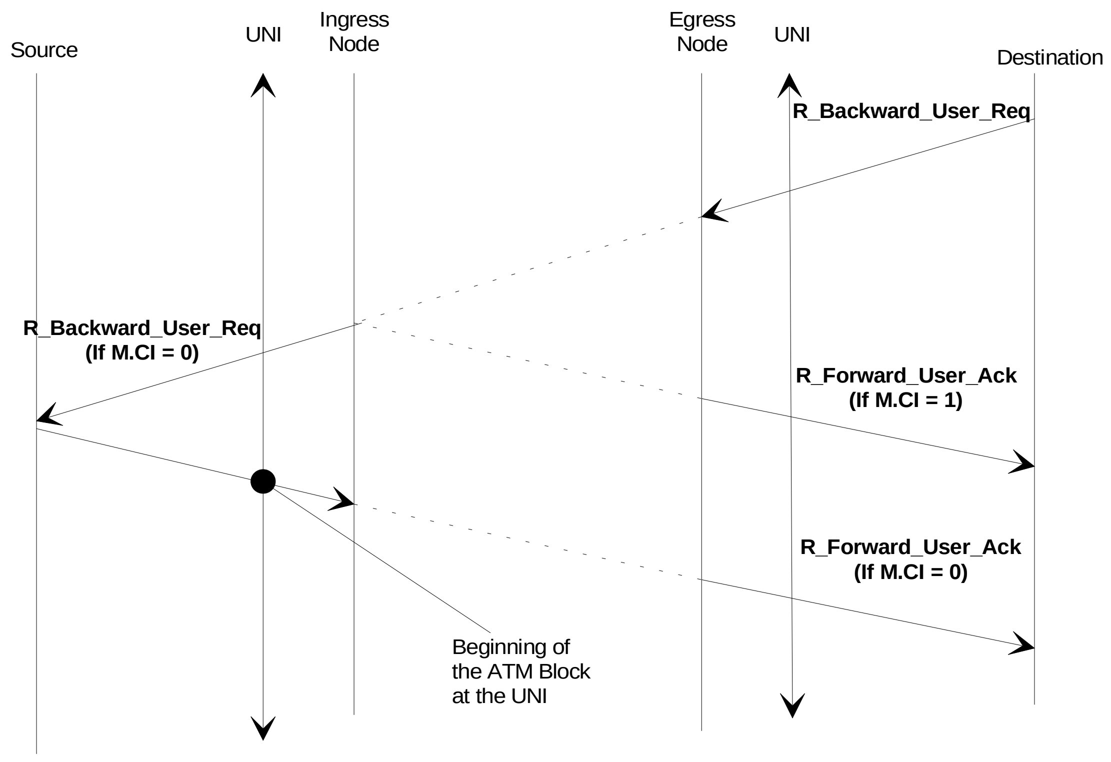
Source
UNI
Ingress Node
Egress Node
UNI
Destination
R_Backward_User_Req
R_Backward_User_Req (If M.CI = 0)
R_Forward_User_Ack (If M.CI = 1)
R_Forward_User_Ack (If M.CI = 0)
Beginning of the ATM Block at the UNI

Figure 4: Rigid Backward BCR Increase or Decrease procedure with no collision

At the Egress Node the requested BCR values are enforced according to (1) and checked to see whether inequality (3) is satisfied. If inequality (3) is satisfied then the request is accepted. If inequality (3) is not satisfied, then the request is processed according to a specific resource allocation policy to check whether the requested resources are available. If this is the case the request is accepted, otherwise it is denied.

For any accepted request, the requested resources are reserved (but not yet allocated) and the R_Backward_User_Req message is forwarded to the next node toward the source with M.CI=0. For any denied request no resources are reserved and the R_Backward_User_Req message is forwarded to the next node toward the source with M.CI=1.

Each node receiving the message checks first the M.CI field: if M.CI=1 the message is forwarded to the next node and no other actions are taken (the bandwidth negotiation request has already been rejected). Otherwise the message is processed to check whether the request can be accepted or not as in the Ingress node and it is then forwarded to the next node.

An R_Backward_User_Req message reaching the Ingress node with M.CI=1 is not forwarded to the source but generates the transmission of an R_Forward_User_Ack message with M.CI=1 back to the destination. The R_Forward_User_Ack message travelling into the network cancels the bandwidth reservation in all the nodes that had been able to reserve the requested resources and it is then forwarded to the destination that does not has any other action to take. The reception of this message terminates the BCR negotiation phase.

An R_Backward_User_Req message reaching the Ingress node with M.CI=0 is processed as usual. If the request can not be accepted the message is not forwarded to the source but an R_Forward_User_Ack message with M.CI=1 is sent back to the destination as in the previous case. Otherwise, the message is forwarded to the source with M.CI=0. The source is then required to send a R_Forward_User_Ack message with M.CI=0 toward the destination. This message delineates a new ATM block and the source shall immediately update its transmission rate after having sent it.

The R_Forward_User_Ack message sent by the source shall carry into the M.Rate(0+1) and M.Rate(OAM) fields BCR values lower or equal to the ones that have been reserved for the connection; these values are enforced at the Ingress node according to (2). Each node receiving this message allocates the reserved resources to the connection and forwards the message downstream. The reception of this message terminates the BCR negotiation phase.

Once the message has reached the Egress node, it is forwarded to the destination and the procedure terminates.

5.6.1.3 Elastic BCR negotiation procedures

5.6.1.3.1 Elastic forward BCR increase

The Elastic Forward BCR Increase procedure is initiated by the source sending an E_Forward_User_Req message.

The flow of messages for this procedure is shown in figure 5.

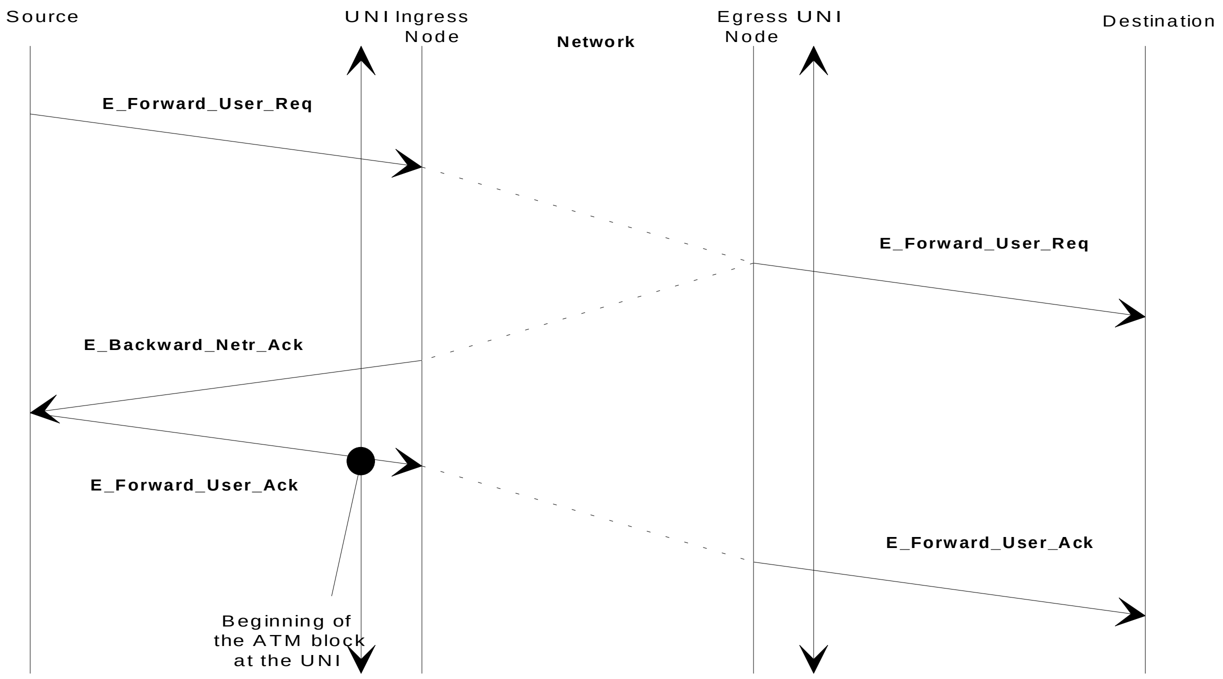

Figure 5: Elastic Forward BCR increase procedure with no collision

The Elastic Forward BCR Increase procedure differs from the corresponding rigid procedure in the following points:

1) the set of messages used by this procedure is the one in table 2 instead of the one in table 1 that are used in the rigid procedure;
2) in the elastic mode the source requests are never denied; each network node updates the M.Rate(0+1) and M.Rate(OAM) fields to indicate the amount of available resources. In particular, the network node may allocate resources between elastic requests in order to achieve some fair share of the available bandwidth. The M.CI field may then be used (i.e., set equal to 1) to indicate that the BCR reserved for the CLP=0+1 cell flow is less than the BCR that would have been obtained under fair share conditions. This is a network option and the definition of fairness is definitely network operator specific. Setting the M.CI field is simply intended to indicate to the source that it could obtain more resources in a next BCR negotiation and to incite downstream network nodes to provision some resources for the connection considered;
3) the E_Forward_User_Req message is always forwarded to the destination, regardless the value of the M.CI field;
4) each node shall either update its internal values of the reserved bandwidth upon the reception of an E_Backward_Net_Ack message (according to the M.Rate(0+1) and M.Rate(OAM) fields) or allocate resources on the basis of the BCR values carried by the E_Forward_User_Ack rather than on the basis of the reserved values.

5.6.1.3.2 Elastic backward BCR increase

The Elastic Backward BCR Increase procedure is initiated by the destination sending an E_Backward_User_Req message.

The flow of messages for this procedure is shown in figure 6.

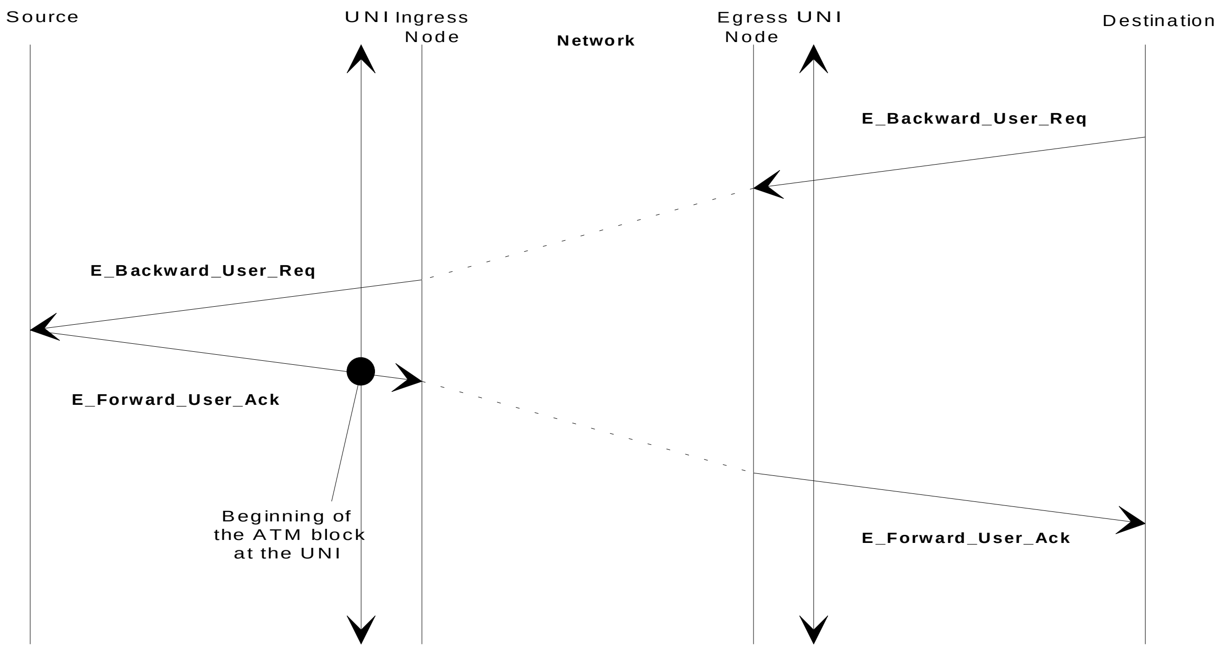

Figure 6: Elastic Backward BCR increase procedure with no collision

The Elastic Backward BCR Increase procedure differs from the corresponding rigid procedure in the following points:

1) the set of messages used by this procedure is the one in table 2 instead of the one in table 1 that are used in the rigid procedure;
2) in the elastic mode the destination requests are never denied; each network node updates the M.Rate(0+1) and M.Rate(OAM) fields to indicate the amount of available resources. In particular, the network node may allocate resources between elastic requests in order to achieve some fair share of the available bandwidth. The M.CI field may then be used (i.e., set equal to 1) to indicate that the BCR reserved for the CLP=0+1 cell flow is less than the BCR that would have been obtained under fair share conditions. This is a network option and the definition of fairness is definitely network operator specific. Setting the M.CI field is simply intended to indicate to the destination that it could obtain more resources in a next BCR negotiation and to incite downstream network nodes to provide some resources for the connection considered;
3) at the reception of an E_Backward_User_Req message the Ingress node never sends an E_Forward_User_Ack message back to the destination, regardless the value of the M.CI field in the received message, but forwards the an E_Backward_User_Req message to the source;
4) each node shall allocate resources on the basis of the BCR values carried by the E_Forward_User_Ack message rather than on the basis of the reserved values.

5.6.2 User maintenance procedures

User Maintenance procedures are identical to Elastic BCR Negotiation procedures apart from the following points:

1) the set of messages used by this procedure is the one in table 3 instead of the one in table 2;
2) maintenance messages do not imply any negotiation of the BCR values allocated to the connection. Upon reception of a Maintenance Request message in a network node, the resource allocation algorithm is disabled and for each cell flow, the corresponding BCR value in the Request message is updated to the minimum of the value contained in Request message and of that allocated in the network node. In other words, the BCR values are updated as:

$$\begin{cases} \text{M. Rate}(0+1) := \min\{\text{M. Rate}(0+1), \Lambda_N(0+1)\}, \\ \text{M. Rate(OAM)} := \min\{\text{M. Rate(OAM)}, \Lambda_N(\text{OAM})\}^{\cdot} \end{cases} \tag{4}$$

3) Maintenance messages shall never have the M.CI field set to 1.

5.6.2.1 Forward status enquiry procedures

The Forward Status Enquiry procedure is identical to the Elastic Forward BCR Increase procedure apart from the points mentioned above. The flow of messages for this procedure is identical to the one shown in figure 5, provided that the corresponding messages of table 3 are substituted to the one shown in that figure.

5.6.2.2 Backward status enquiry procedures

The Backward Status Enquiry procedure is identical to the Elastic Backward BCR Increase procedure apart from the points mentioned above. The flow of messages for this procedure is identical to the one shown in figure 6, provided that the corresponding messages of table 3 are substituted to the one shown in that figure.

5.6.3 Traffic control procedures

Traffic Control procedures are similar to BCR negotiation procedures, with the difference that they are initiated by the Ingress node or the Egress node instead of by the source or the destination. In addition, Traffic Control messages shall have the M.Number field properly set, since Traffic Control procedures make use of this field.

5.6.3.1 Rigid traffic control procedures

5.6.3.1.1 Rigid forward traffic control

A Rigid Forward Traffic Control procedure is initiated by the Ingress node by sending into the forward direction an R_Traffic_Forward_Req. The flow of messages for this procedure is shown in figure 7.

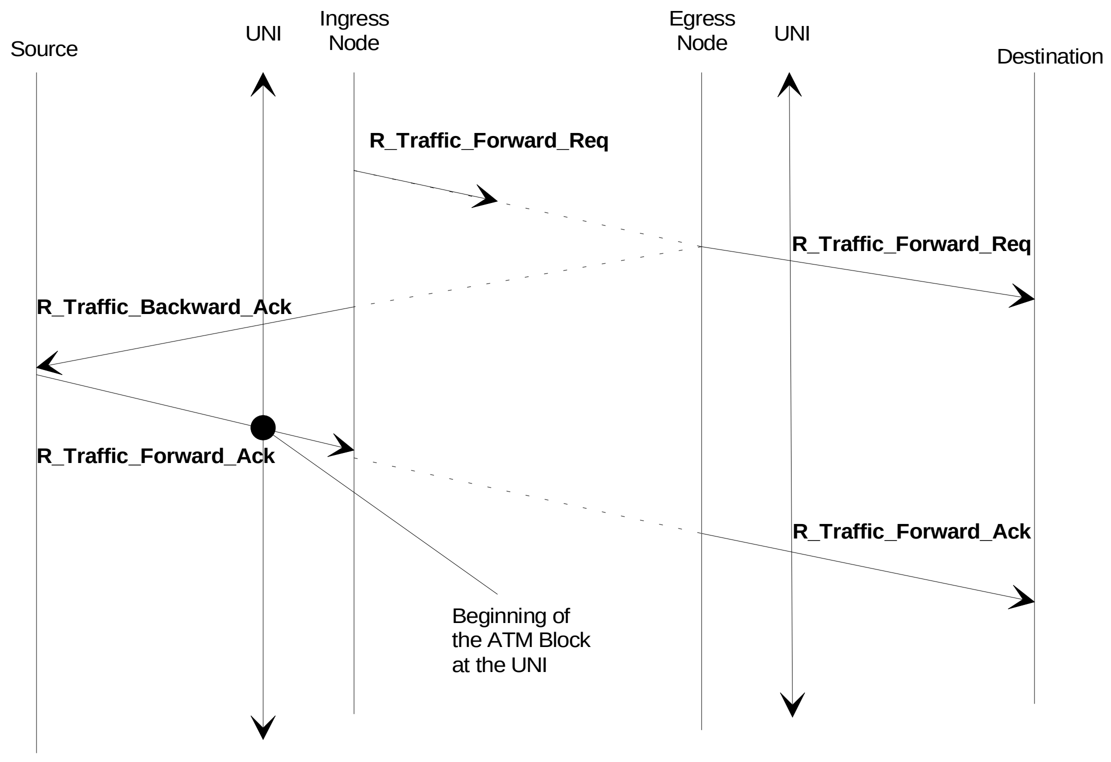

Figure 7: Rigid Forward Traffic Control procedure with no collision

The R_Traffic_Forward_Req message sent by the Ingress node shall have M.Rate (0+1)= $\Lambda_{current}$(0+1), M.Rate(OAM)= $\Lambda_{Current}$(OAM), and M.Number set to the proper progressive number.

Upon reception of this message in a network node, the requested resources are examined. If they are not available according to the resource allocation algorithm, the M.CI field is set and a possible BCR for the CLP=0+1 cell flow is computed. This new value is overwritten in the M.Rate(0+1) field.

Once received at the Egress node, the request message is forwarded to the destination. In addition, the Egress node shall send back a R_Traffic_Backward_Ack message having the same values of the fields M.CI, M.Rate(0+1) and M.Rate(OAM). Upon reception of this message at the Ingress node, the Ingress node computes the new BCR values to be proposed to the source, writes them in the M.Rate(0+1) and M.Rate(OAM) fields and forwards the message to the source. If M.CI=1, these BCR values depend on the policing strategy adopted by the network. For instance, they may be equal to 0, to minimum BCR values, or to the possible BCR values computed in the network nodes. In the contrary, if M.CI=0 the BCR values are exactly the original values $\Lambda_{current}(0+1)$ and $\Lambda_{current}(OAM)$.

The R_Traffic_Backward_Ack message should itself be acknowledged by the source by sending an R_Traffic_Forward_Ack message which is the leading RM cell of a new ATM block. The BCR values acknowledged by the source should be less than or equal to the BCR values proposed by the network and are enforced at the Ingress node according to (2). Upon the reception of this message each network node modifies the resources allocated to the connection according to the BCR values carried in the message and forwards the message to the downstream node. Once the message is received by the destination the procedure terminates.

5.6.3.1.2 Rigid backward traffic control

A Rigid Backward Traffic Control procedure is initiated by the Egress node by sending into the backward direction an R_Traffic_Backward_Req.

The flow of messages for this procedure is shown in figure 8.

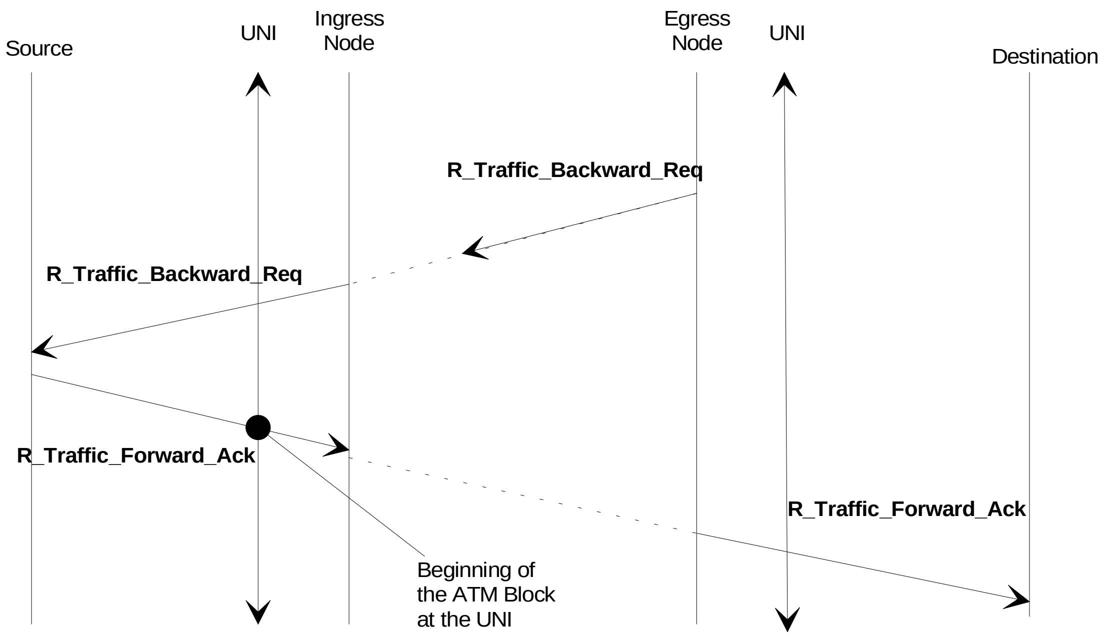

Figure 8: Rigid Backward Traffic Control procedure with no collision

The R_Traffic_Backward_Req message sent by the Egress node shall have M.Rate (0+1)= $\lambda_{current}(0+1)$, M.Rate(OAM)= $\lambda_{Current}(OAM)$, and M.Number set to the proper progressive number.

This request message is handled as the corresponding request message in the Rigid Forward Traffic Control procedure described in subclause 5.6.3.1.1 (computation of possible BCR values in each network node). In particular, if the requested resources are not available, the M.CI field is set.

Upon reception of this R_Traffic_Backward_Req message at the ingress node, if M.CI=0 the BCR $\Lambda_{Current}(0+1)$ and $\Lambda_{Current}(OAM)$ are updated according to the values of the M.Rate(0+1) and M.Rate(OAM) fields and the message is forwarded to the source. Note that for setting the M.CI field, the Ingress node should take into account the BCR parameters of the UPC. If M.CI=1, the R_Traffic_Backward_Req message is forwarded to the source with some proposed BCR values which depend on the network policing strategy. Upon reception of the R_Traffic_Backward_Req message the source shall send a R_Traffic_Forward_Ack message with the same value of the M.CI field. This message delineates a new ATM block. The BCR values conveyed by this message should be less than or equal to the values suggested by the Ingress node and are enforced at the Ingress node according to (2). Upon reception of this message each network node modifies the resources allocated to the connection according to the BCR values carried in the message and forward the message to the downstream node. Once the message is received by the destination the procedure terminates.

5.6.3.2 Elastic traffic control procedures

5.6.3.2.1 Elastic forward traffic control

An Elastic Forward Traffic Control procedure is initiated by the Ingress node by sending into the forward direction a R_Traffic_Forward_Req.

The flow of messages for this procedure is shown in figure 9.

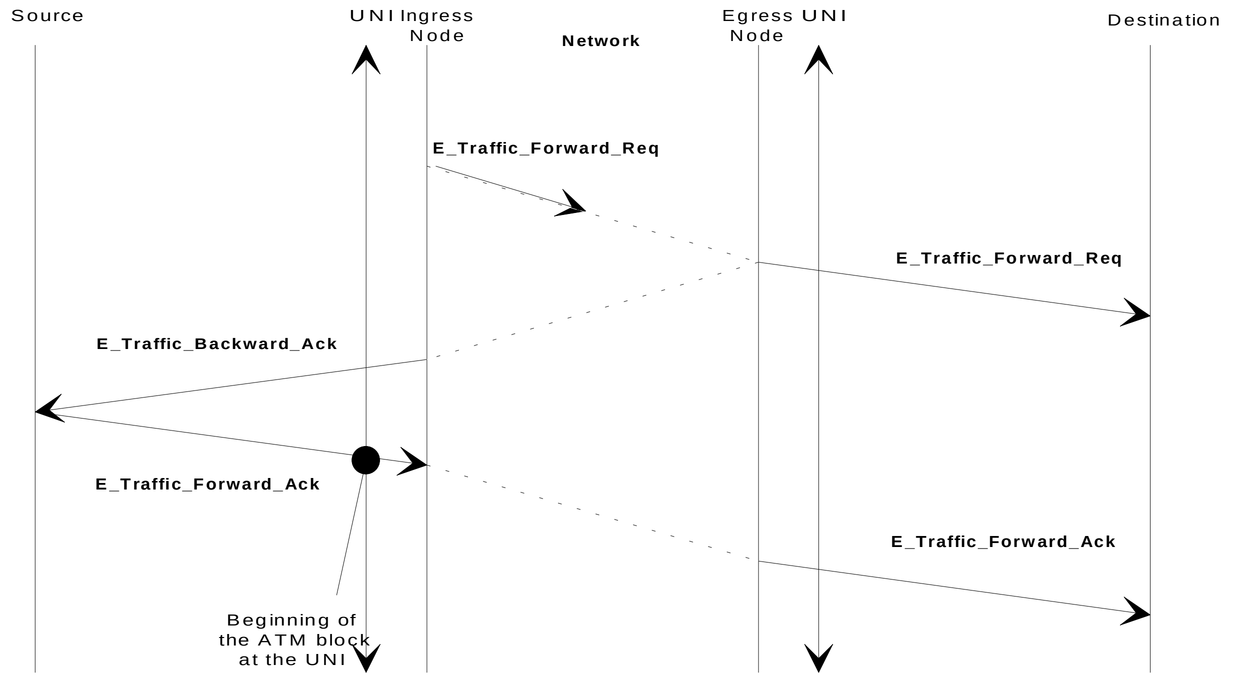

Figure 9: Elastic Forward Traffic Control procedure with no collision

In the elastic mode, the BCR values contained in the R_Traffic_Forward_Req message transmitted from the Ingress node may be greater than the BCR values currently allocated but shall be less than or equal to the values $\Lambda_{max}(0+1)$ and $\Lambda_{max}(OAM)$ negotiated at connection set up. Upon reception of this message, some resources are reserved in the network nodes by applying the resource allocation algorithm as for a usual BCR negotiation in the elastic mode (also the M.CI field is handled in the same way).

Once the request message has been processed at the Egress node, it is forwarded to the destination for information. In addition, the Egress node shall send back toward the source an E_Traffic_Backward_Ack message with the same values of the M.CI, M.Rate(0+1) and M.Rate(OAM) fields. Each node, receiving this message, may update its internal values of the reserved bandwidth accordingly.

Upon reception of this message at the Ingress node, this message is forwarded to the source with the proposed BCR values for the CLP=0+1 and OAM cell flows. The source shall acknowledge this message by sending an E_Traffic_Forward_Ack message that delineates a new ATM block. The acknowledged BCR values should be less than or equal to those proposed by the network and are enforced at the Ingress node according to (2). At the reception of this message each network node allocates the requested resources to the connection and forwards the message to the downstream node. Once the message is received by the destination the procedure terminates.

5.6.3.2.2 Elastic backward traffic control

An Elastic Backward Traffic Control procedure is initiated by the Egress node by sending into the backward direction an R_Traffic_Backward_Req.

The flow of messages for this procedure is shown in figure 10.

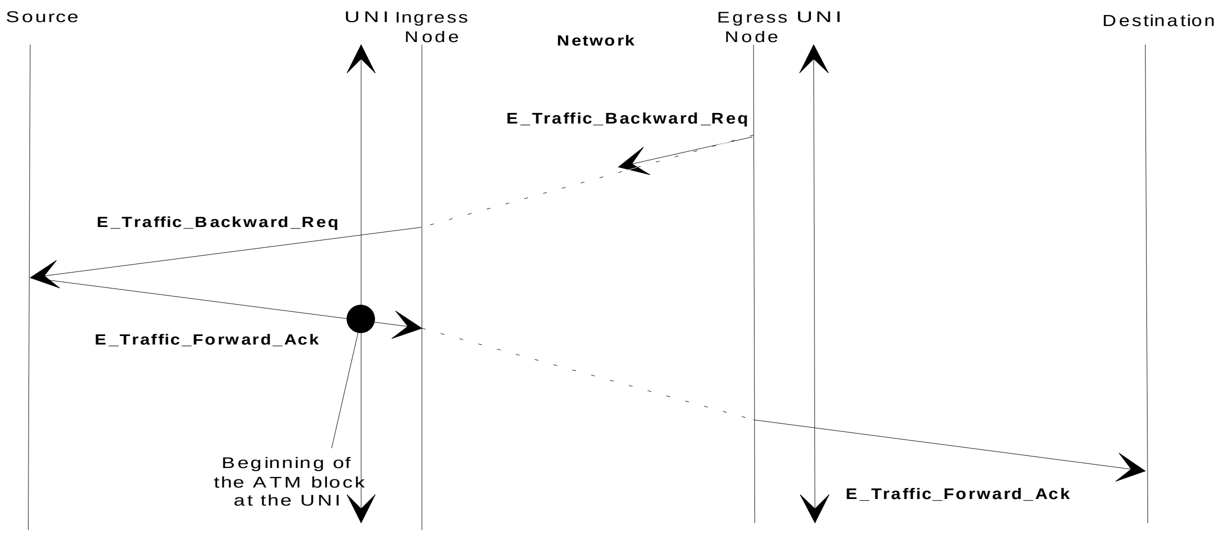

Figure 10: Elastic Backward Traffic Control procedure with no collision

In the elastic mode, the BCR values contained in the E_Traffic_Backward_Req message transmitted from the Egress node may be greater than the BCR values currently allocated at network egress but shall be less than or equal to the values $\Lambda_{max}(0+1)$ and $\Lambda_{max}(OAM)$ negotiated at connection set up. Upon reception of this message, some resources are reserved in the network nodes by applying the resource allocation algorithm as for a usual BCR negotiation in the elastic mode (also the M.CI field is handled in the same way).

Once this request message has been processed at the Ingress node, it is forwarded to the source with the proposed BCR values for the CLP=0+1 and OAM cell flows. The source shall acknowledge this message by sending an E_Traffic_Forward_Ack that delineates a new ATM block. The acknowledged BCR values should be less than or equal to the BCR values proposed by the network and are enforced at the Ingress node according to (2). At the reception of this message each network node allocates the requested resources to the connection and forwards the message to the downstream node. Once the message is received by the destination the procedure terminates.

5.6.4 Network maintenance procedures

5.6.4.1 Time-out handling procedures

5.6.4.1.1 Timers associated with ABT/DT procedures

Some protocol messages (either Bandwidth Request or Bandwidth Acknowledgement messages) may be lost within the network. In that case, after initiating a given ABT/DT procedure, it may happen that a given entity (the source, the destination, a network node, etc.) does not get any response from the network or either user. To overcome the problems due to the loss of messages or to the absence of response from an entity, some timers are introduced as follows:

1) **W_Source:** after initiating a BCR negotiation, the source should get a response from the network within a time interval at most W_Source time unit long. Timer W_Source is activated each time the source initiates an ABT/DT procedure (i.e., immediately after having sent a R_Forward_User_Req message, or an E_Forward_User_Req message or a M_Forward_User_Req message); the timer is deactivated at the reception of one of the following messages:

   - the backward acknowledgement message corresponding to the forward request message sent by the source (i.e. a R_Backward_Net_Ack message or an E_Backward_Net_Ack message or a M_Backward_Net_Ack message); this happens in case of normal completion of the procedure;

   - an M_Backward_Notready_Ack; this happens in the case of collision of the request at the Ingress node (see subclause 5.7.1). Deactivation of the timer W_Source upon reception of this message is an implementation option that does not affect the behaviour of the protocol;

   - an R_Backward_User_Req message or an E_Backward_User_Req message; this happens in case of collision within the network with a backward BCR negotiation request;

- an R_Traffic_Backward_Req message or an E_Traffic_Backward_Req message; this happens in case of collision within the network with a backward Traffic Control request;
- an N_Backward_Ack message; this happens in case of expiration of timer W_Ingress (see subclause 5.6.4.1.2).

Actions taken by the source in case timer W_Source expires are implementation specific.

2) **W_Destination:** similarly to the source, after initiating a BCR negotiation, the destination should get a response from the network within a time interval at most W_Destination time unit long. Timer W_Destination is activated each time the destination initiates an ABT/DT procedure (i.e., immediately after having sent a R_Backward_User_Req message, or an E_Backward_User_Req message or a M_Backward_User_Req message); the timer is deactivated upon reception of one of the following messages:

– the forward acknowledgement message corresponding to the backward request message sent by the source (i.e. a R_Forward_User_Ack message or an E_Forward_User_Ack message or a M_Forward_User_Ack message); this happens in the case of a normal completion of the procedure;

- an M_Forward_Notready_Ack; this happens in the case of collision of the request at the Egress node (see subclause 5.7.2). Deactivation of the timer W_Destination upon reception of this message is an implementation option that does not affect the behaviour of the protocol;
- an R_Traffic_Forward_Req message or an E_Traffic_Forward_Req message; this happens in the case of collision within the network with a forward Traffic Control request;
- an N_Forward_Ack message; this happens in case of expiration of timer W_Egress (see subclause 5.6.4.1.2).

Actions taken by the destination in case timer W_Destination expires are implementation specific.

3) **W_Ingress:** after sending a Request message (either a Bandwidth Request or a Traffic Control Request) into the network, the Ingress node should get an answer within a time interval at most W_Ingress time units long. Timer W_Ingress is activated at the Ingress node immediately after the transmission of an R_Forward_User_Req message, or an E_Forward_User_Req message or a R_Traffic_Forward_Req message or an E_Traffic_Forward_Req message or an N_Forward_Req message; the timer is deactivated upon reception of one of the following messages:

- an R_Backward_Net_Ack message or an _E_Backward_Net_Ack message; this happens if the timer was previously activated by a R_Forward_User_Req message or an E_Forward_User_Req message and no collisions occur neither with a backward BCR negotiation request nor with a backward Traffic Control request;
- an R_Backward_User_Req message or an E_Backward_User_Req message; this happens if the timer was previously activated by an R_Forward_User_Req message or an E_Forward_User_Req message and a collision occurs with a backward BCR negotiation request but not with a backward Traffic Control request;
- an R_Traffic_Backward_Req message or an E_Traffic_Backward_Req message; this happens if the timer was previously activated by an R_Forward_User_Req message or an E_Forward_User_Req message and a collision occurs with a backward Traffic Control request initiated by the destination;
- an R_Traffic_Backward_Ack message or an E_Traffic_Backward_Ack message; this happens if the timer was previously activated by a forward Traffic Control request;
- a N_Backward_Ack message; this happens if the timer was previously activated by a N_Forward_Req message.

In case timer W_Ingress expires the ingress node initiates an Ingress Time out Recovering procedure as described in subclause 5.6.4.1.2.

4) **W_Egress:** similarly to what happens at the ingress node, after sending a Request message (either a Bandwidth or a Traffic Control Request) into the network the Egress node should get an answer within a time interval at most W_Egress time units long. The same principle holds after the transmission of a backward acknowledgement cell. Summarizing, timer W_Egress is activated at the Egress node immediately after the transmission of one of the following messages:

- an R_Backward_User_Req message or an E_Backward_User_Req message;
- an R_Traffic_Backward_Req message or an E_Traffic_Backward_Req message;
- an N_Backward_Req message;
- an R_Backward_Net_Ack message with M.CI=0;
- an E_Backward_Net_Ack message;
- an M_Backward_Net_Ack message;
- an R_Traffic_Backward_Ack message or an E_Traffic_Backward_Ack message.

The timer is deactivated at the reception of one of the following messages:

- an R_Forward_User_Ack message or an E_Forward_User_Ack message; this happens if the timer was previously activated by a backward BCR request message or by a backward BCR acknowledgement message;
- an R_Traffic_Forward_Req message or an E_Traffic_Forward_Req message; this happens if the timer was previously activated by a backward BCR request message or by a backward Traffic Control request message and a collision occurs with a forward Traffic Control request;
- an R_Traffic_Forward_Ack message or an E_Traffic_Forward_Ack message; this happens if the timer was previously activated by a backward Traffic Control request and no collisions occur with forward Traffic Control Request;
- an N_Forward_Ack message; this occurs if the timer was previously activated by an N_Backward_Req message.

In the case timer W_Egress expires the egress node initiates an Egress Time out Recovering procedure as described in subclause 5.6.4.1.3.

Note that Traffic Control messages should have correct Sequence Number in order to be taken into account at the Egress or Ingress node. Traffic Control messages with incorrect Sequence Numbers are ignored.

The meaning of the above timers is illustrated in figure 11 for the case of a Rigid Forward BCR Negotiation and in figure 12 for the case of a Rigid Backward BCR Negotiation.

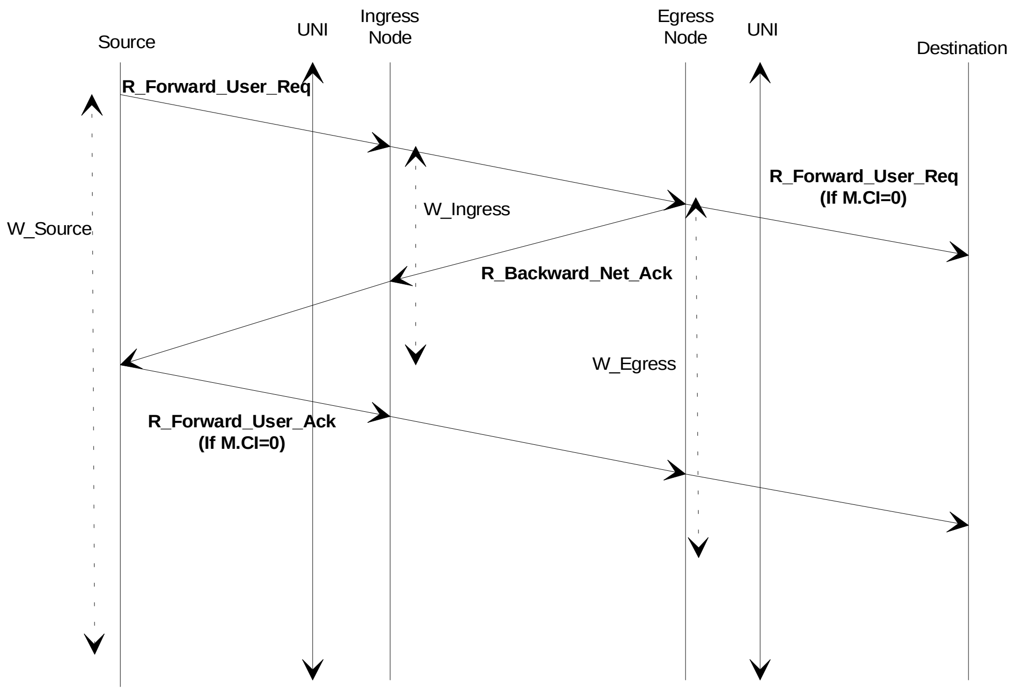

Figure 11: Timer associated with a Rigid Forward BCR negotiation

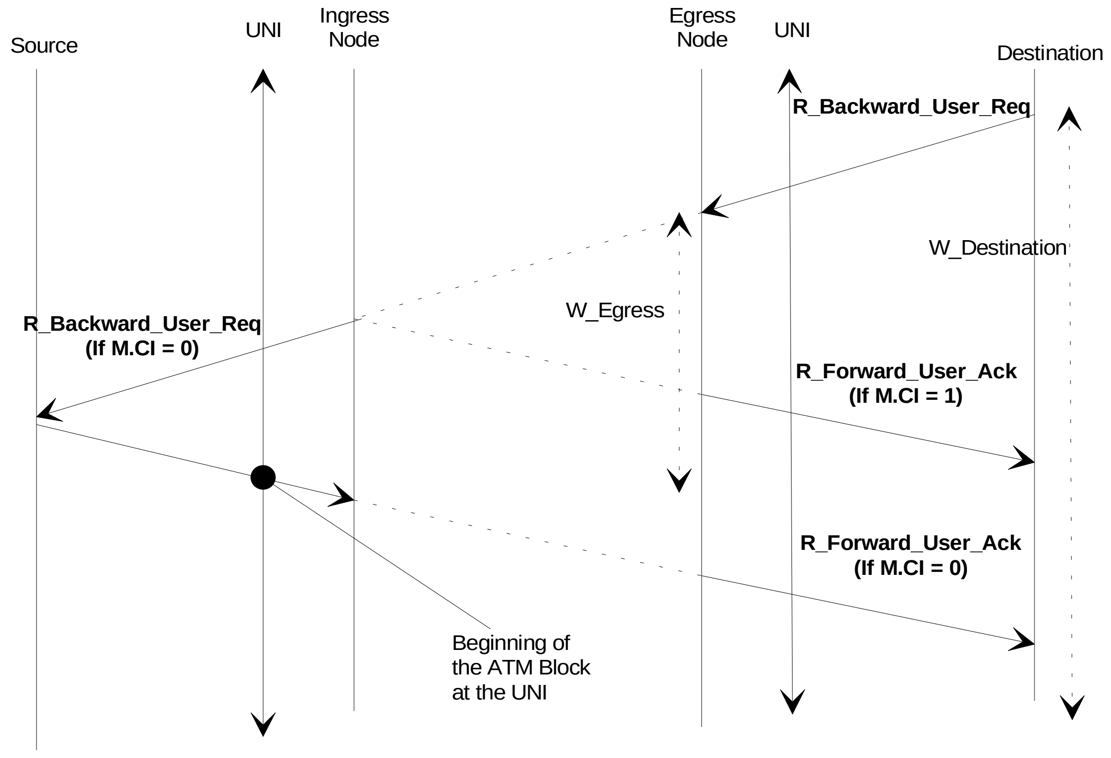
Source
UNI
Ingress Node
Egress Node
UNI
Destination
R_Backward_User_Req
W_Destination
W_Egress
R_Backward_User_Req (If M.CI = 0)
R_Forward_User_Ack (If M.CI = 1)
R_Forward_User_Ack (If M.CI = 0)
Beginning of the ATM Block at the UNI

Figure 12: Timer associated with a Rigid Backward BCR negotiation

5.6.4.1.2 Recovering from ingress Time-Out

Let us consider Rigid BCR negotiations processed at network ingress. The regular case (i.e. no RM cell loss and no collision with a BCR negotiation by the destination) is depicted in figure 11.

If one of the relevant messages does not arrive at network ingress within a time interval at most W_Ingress long, the network retries the transaction. Specifically, an N_Forward_Req message with an appropriate Sequence Number is sent into the network with the BCR $\Lambda_{Current}(0+1)$ and $\Lambda_{Current}(OAM)$ currently allocated at network ingress if no BCR values are reserved or $\Lambda'_{Current}(0+1)$ and $\Lambda'_{Current}(OAM)$ if some BCR values have been reserved; the Elastic bit is set equal to that of the transaction which is pending. Timer W_Ingress is reinitialised after the transmission of this message. Such a message has priority over any BCR negotiation messages sent by the destination or Traffic Control message coming from the network egress.

Upon reception of this message in a network node, a possible BCR for the CLP=0+1 cell flow is computed and reserved according to the value of the Elastic bit as for any other transaction; the elastic bit is handled accordingly.

At network egress, the N_Forward_Req message is forwarded to the destination for information and a N_Backward_Ack message is sent back to network ingress; timer W_Egress is deactivated, if necessary, and (re)initialized. Upon reception of this N_Backward_Ack message at network ingress, timer W_Ingress is deactivated and the message is forwarded to the source to indicate that some BCR negotiation message have been lost within the network and that the BCR values $\Lambda_{Alloc}(0+1)$ and $\Lambda_{Alloc}(OAM)$ are to be allocated within the network. The source should acknowledge the N_Backward_Ack message by sending back the N_Forkward_Ack message by itself. The BCR $\Lambda_{Alloc}(0+1)$ and $\Lambda_{Alloc}(OAM)$ are allocated to the connection upon reception of this message . Timer W_Egress is deactivated upon reception of one of the above messages, which is forwarded to the destination. The actions taken by the network when the timer W_Ingress expires are shown in figure 13.

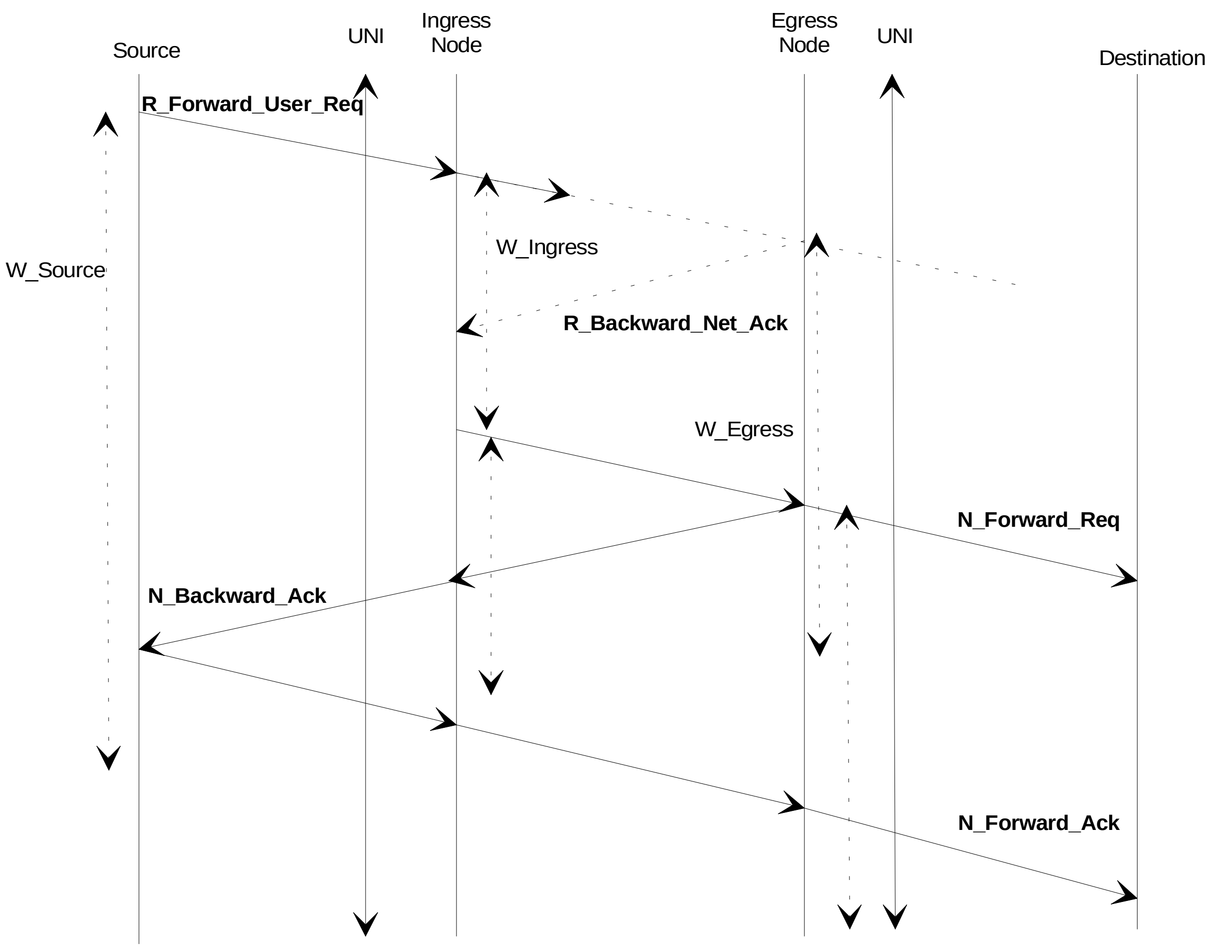

Figure 13: Recovering from W_Ingress Time-Out

5.6.4.1.3 Recovering from Egress Time-Out

Besides timer W_Ingress, it may happen that timer W_Egress expires. In the case of a forward BCR negotiation this timer is activated when the R_Backward_Net_Ack (or E_Backward_Net_Ack) message is sent back to the source, as shown in figure 11. Timer W_Egress is activated also in the case of a backward BCR negotiation when the R_Backward_User_Req (or E_Backward_User_Req) message is sent into the network, as shown in figure 12. In all these cases, the appropriate Acknowledgement message Bandwidth Acknowledgement RM cell is expected at network egress within a given time interval. If this message is not received before timer W_Egress expire, special actions intended to retry the incomplete transaction are taken at network egress. Specifically, an N_Backward_Req message with an appropriate Sequence Number is sent on the backward direction into the network. The BCR values conveyed by this message are the values $\lambda_{Current}(0+1)$ and $\lambda_{Current}(OAM)$ currently allocated to the CLP=0+1 and OAM cell flow at network egress if no BCR values have been reserved or $\lambda'_{Current}(0+1)$ and $\lambda'_{Current}(OAM)$ if some BCR values have been reserved; the elastic bit in this message is set equal to that of the transaction, which is pending.

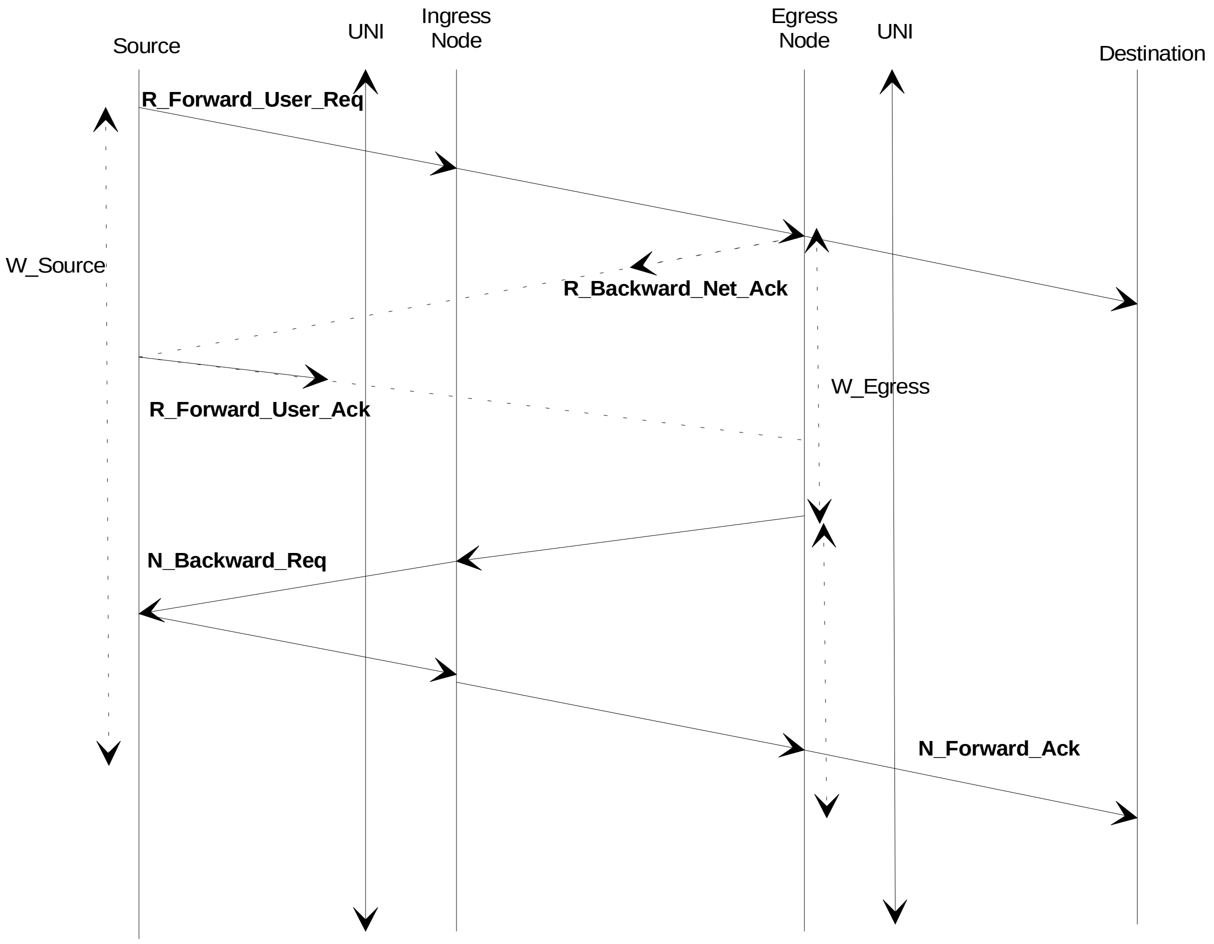

Figure 14: Recovering from W_Egress Time-Out

Upon reception of this message in a network node, a possible BCR for the CLP=0+1 cell flow is computed by taking into account the Elastic bit as for any other transaction and the Elastic bit is set accordingly.

At network ingress, upon reception of the N_Backward_Req message, the message is forwarded to the source and no other actions are taken by the ingress node.

The source should acknowledge the N_Backward_Req message by itself by sending a N_Forward_Ack message, which is then forwarded to the network egress.

Upon reception of the N_Forward_Ack message in a network node, the BCR $\Lambda_{Alloc}(0+1)$ is updated to the value conveyed in the message. The traffic management process in the case of expiration of timer W_Egress is shown in figure 14 for the case in which timer W_Egress has been triggered by the transmission of a R_Backward_Net_Ack message (rigid forward BCR negotiation procedure). The other cases of expiration of timer W_Egress are dealt with in the same way.

5.6.4.1.4 Further considerations

The above procedures related the expiration of timers W_Ingress and W_Egress may be reiterated several times. Beyond a certain number of iterations, the network may declare the connection as to be non compliant and release the connection. The definition of non compliant connection is network operator specific.

If the source (the destination) has not received any response from the network within a time interval at most W_Source (W_Destination) time unit long after having initiated a BCR negotiation, the source (the destination) might reattempt the BCR negotiation.

5.7 Behaviour in the case of collision of different procedures

Collisions may happen at ingress node, egress node or within the network; the procedures followed in these three cases are different.

5.7.1 Collision at the ingress node

Collision at the ingress node occurs when the source tries to initiate a new forward BCR negotiation (or a forward Status Enquiry procedure) when another one is still in progress within the network or when the network has initiated a Forward Traffic Control procedure. In both the cases the new negotiation request is denied and a M_Backward_Notready_Ack message is sent to the source, as shown in figure 15.

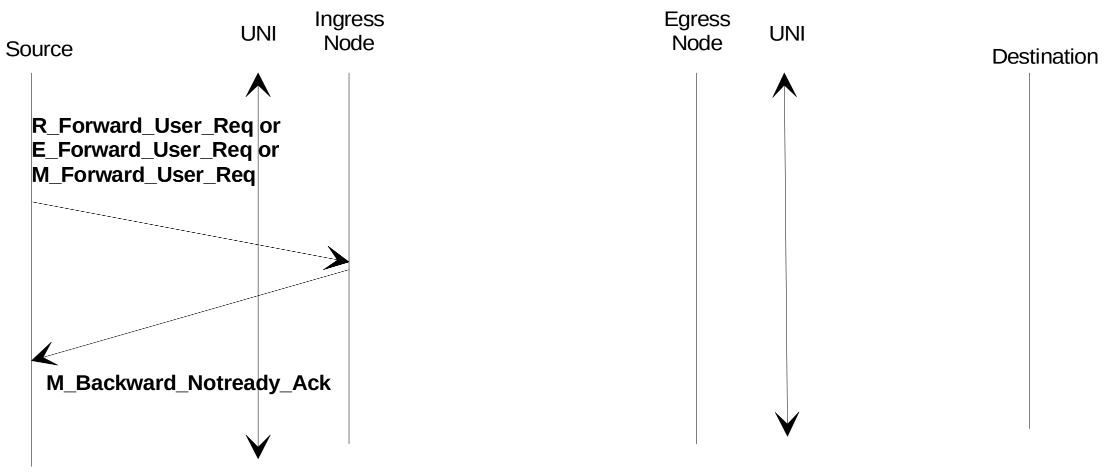

Figure 15: Collision at the ingress node

5.7.2 Collision at the egress node

Collision at the egress node occurs when the destination tries to initiate a new backward BCR negotiation (or a backward Status Enquiry procedure) when another one is still in progress within the network or when the network has initiated a Backward Traffic Control procedure. In both cases the new negotiation request is denied and a M_Forward_Notready_Ack message is sent to the source, as shown in figure 16.

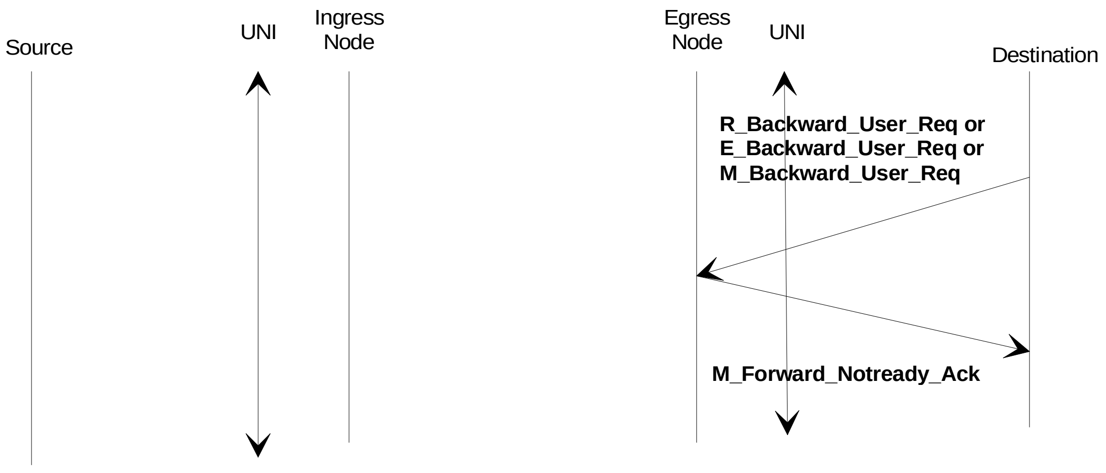

Figure 16: Collision at the Egress node

5.7.3 Collision within the network

Since BCR negotiations in a given direction may be initiated by either user, two BCR negotiations may collide within the network. Specifically, two BCR negotiations collide if the Forward BCR Negotiation Request message issued by the source arrives at a network node, which has received a Backward BCR Negotiation Request message from the destination and not the corresponding Acknowledgement message. To solve this problem, BCR negotiations initiated by the destination have priority over BCR negotiation initiated by the source (backward priority principle).

Furthermore, Traffic Control messages (with M.Maintenance=0 and M.Trf=1) issued either at network egress or ingress in the case of potential congestion have priority over any BCR negotiation initiated by the source or the destination. Finally, those traffic control messages issued at network ingress have priority over those generated at network egress.

Summarizing, the following priority order applies (decreasing priority):

1) forward Traffic Control procedures;

2) backward Traffic Control procedures;

3) backward BCR negotiation procedures;

4) forward BCR negotiation procedures.

5.7.3.1 Collision between forward and backward BCR negotiation

Let consider first the case in which a Forward BCR Negotiation collides with a Backward BCR Negotiation but no Traffic Control procedures are initiated by the network. In that case the Backward BCR Negotiation procedure takes priority over the Forward BCR Negotiation procedure and will be completed in the normal way (see subclauses 5.6.1.2.3 and 5.6.1.3.2) whereas the Forward BCR Negotiation will be aborted. Upon reception at a network node of a Backward Bandwidth Request message due to the destination, any BCR reservation by the source is cancelled. On the other hand any Forward Bandwidth Request message issued by the source and arriving at a network node while some resources have been reserved by a Backward Bandwidth Request message issued by the destination is ignored by the network node and the M.CI field is set, if necessary. No acknowledgement to this request is sent at network egress to the source.

(In replacement of editor’s note).<u>The only difference with the no collision case is that in a rigid backward BCR negotiation the R_Backward_User_Req is forwarded to the source even if M.CI=1. This is intended to alert the source that a collision happened in the network. The source has to acknowledge this message only if M.CI=0 or M.Elastic=0.</u>

The messages exchanged across the UNI are shown in figures 17 and 18 in case of Rigid and Elastic Backward BCR negotiations respectively.

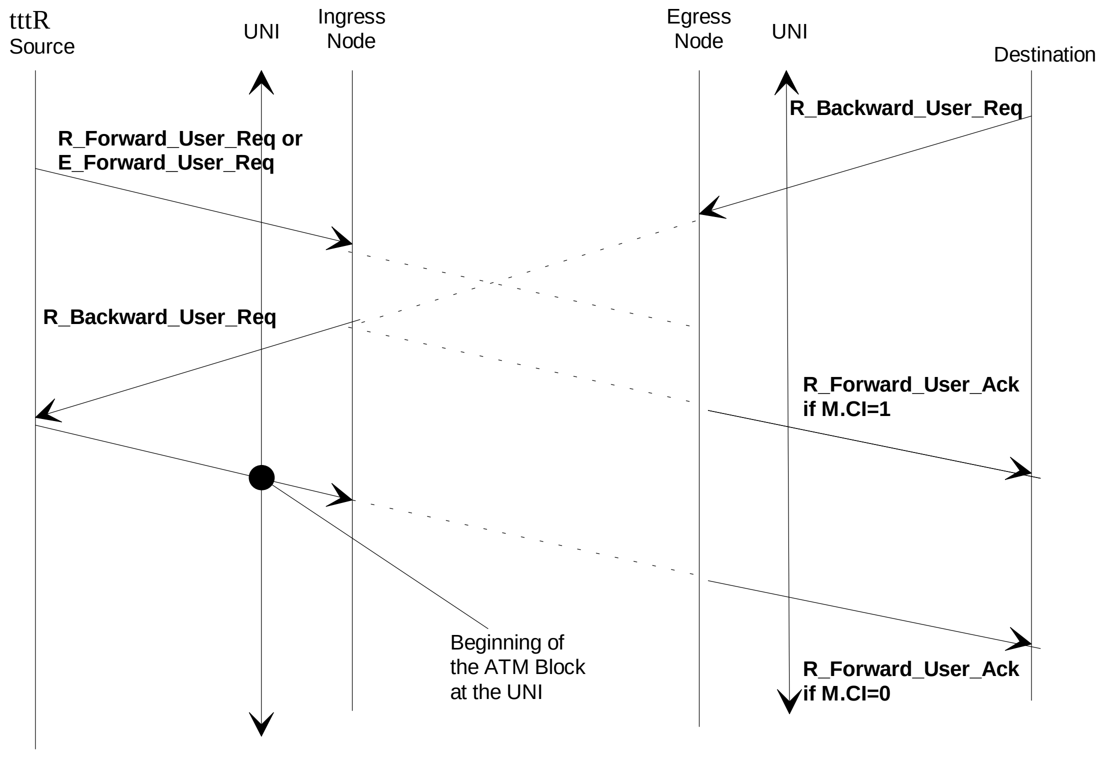
tttR
Source
UNI
Ingress Node
Egress Node
UNI
Destination
R_Backward_User_Req
R_Forward_User_Req or E_Forward_User_Req
R_Backward_User_Req
R_Forward_User_Ack if M.CI=1
Beginning of the ATM Block at the UNI
R_Forward_User_Ack if M.CI=0

Figure 17: BCR negotiation initiated by the destination in the rigid mode with collision

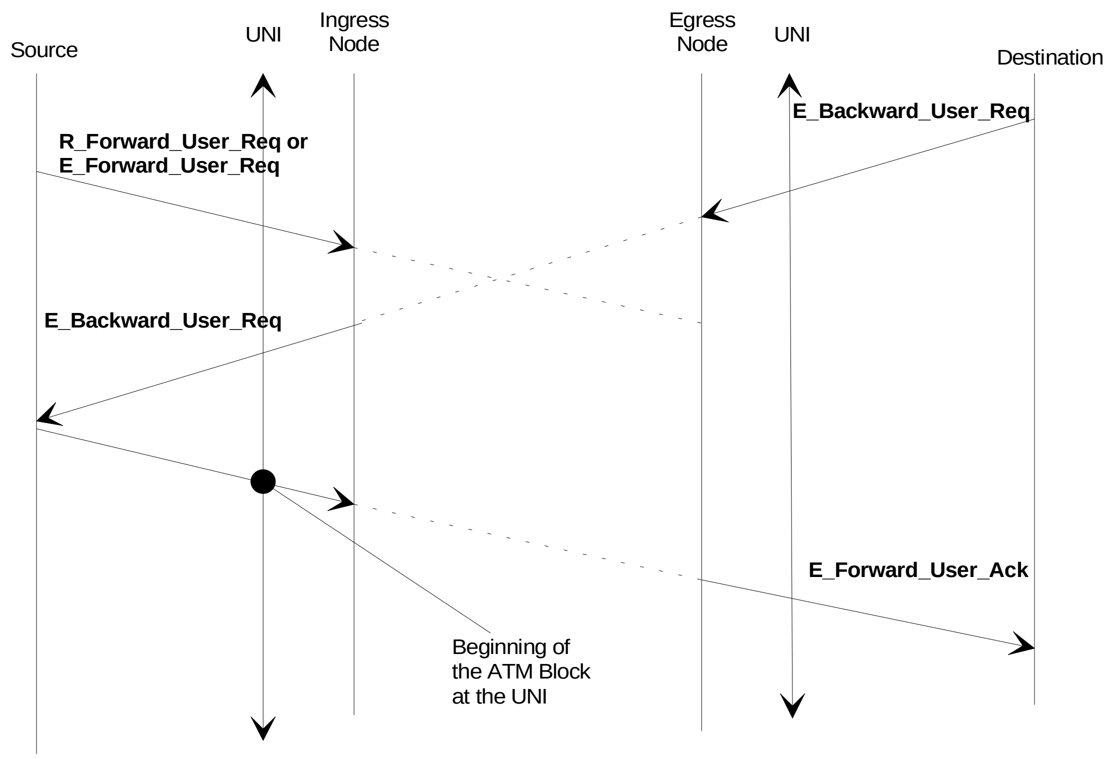

Figure 18: BCR negotiation initiated by the destination in the elastic mode with collision

5.7.3.2 Collision between backward BCR negotiation and Forward Traffic Control procedures

When a Backward BCR Negotiation collides with a Forward Traffic Control procedure the latter takes priority and is completed in the normal way (see subclauses 5.6.3.1.1 and 5.6.3.2.1) whereas the Backward BCR Negotiation is aborted. Upon reception at a network node of a Forward Traffic Control Request message, any BCR reservation made by the destination is cancelled. On the other hand any Backward Bandwidth Request message issued by the destination and arriving at a network node while some resources have been reserved by a Forward Bandwidth Request message is ignored by the network node and the M.CI field is set, if necessary. No acknowledgement to this request is sent to the destination. The messages exchanged across the UNI are shown in figures 19 and 20 in case of Rigid and Elastic Forward Traffic Control, respectively.

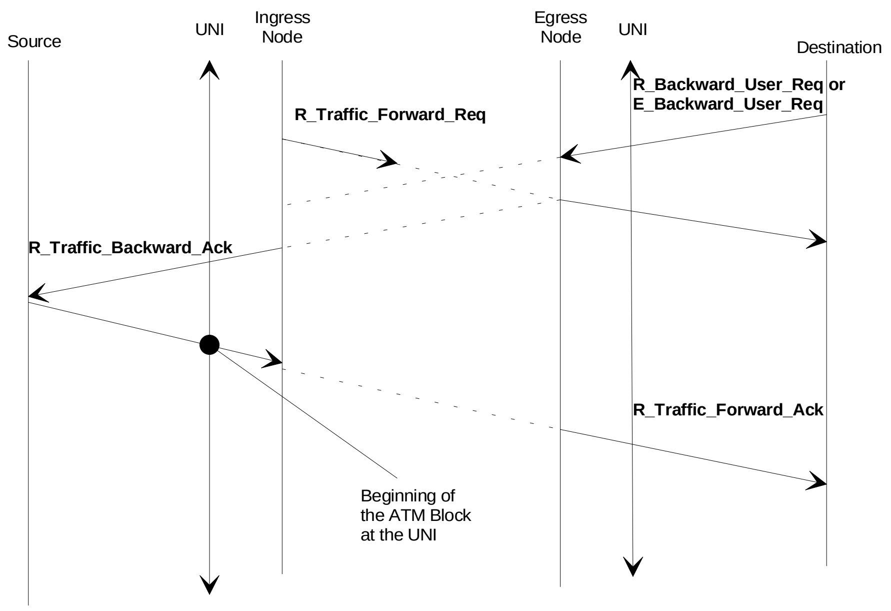

Figure 19: Forward Traffic Control procedure in the rigid mode with collision with a Backward BCR Negotiation procedure

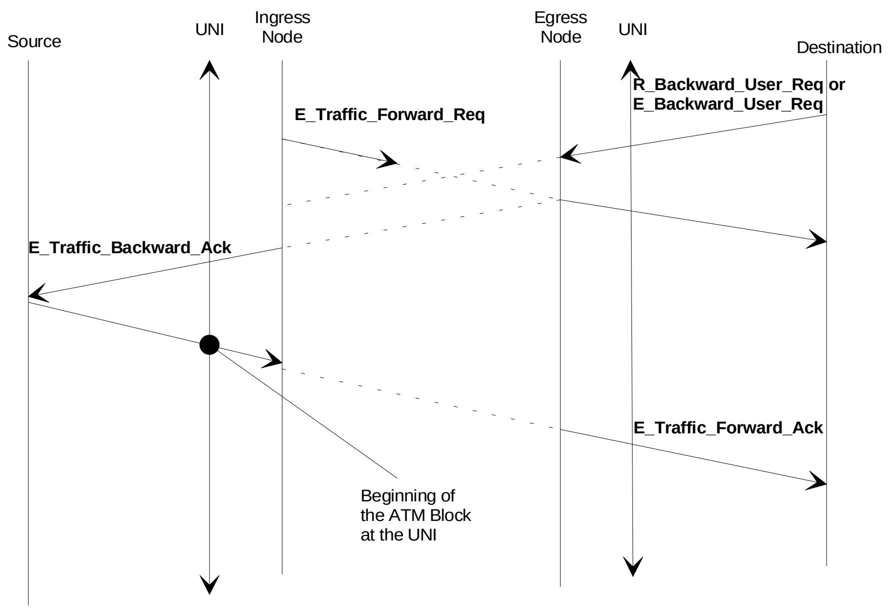

Figure 20: Forward Traffic Control procedure in the elastic mode with collision with a Backward BCR negotiation procedure

5.7.3.3 Collision between forward BCR negotiation and backward traffic control procedures

When a Forward BCR Negotiation collides with a Backward Traffic Control procedure the latter takes priority and is completed in the normal way (see subclauses 5.6.3.1.2 and 5.6.3.2.2) whereas the Backward BCR Negotiation is aborted. Upon reception at a network node of a Backward Traffic Control Request message, any BCR reservation made by the source is cancelled. On the other hand any Forward Bandwidth Request message issued by the source and arriving at a network node while some resources have been reserved by a Backward Bandwidth Request message is ignored by the network node and the M.CI field is set, if necessary. No acknowledgement to this request is sent to the source. The messages exchanged across the UNI are shown in figures 21 and 22 in the case of Rigid and Elastic Backward Traffic Control, respectively.

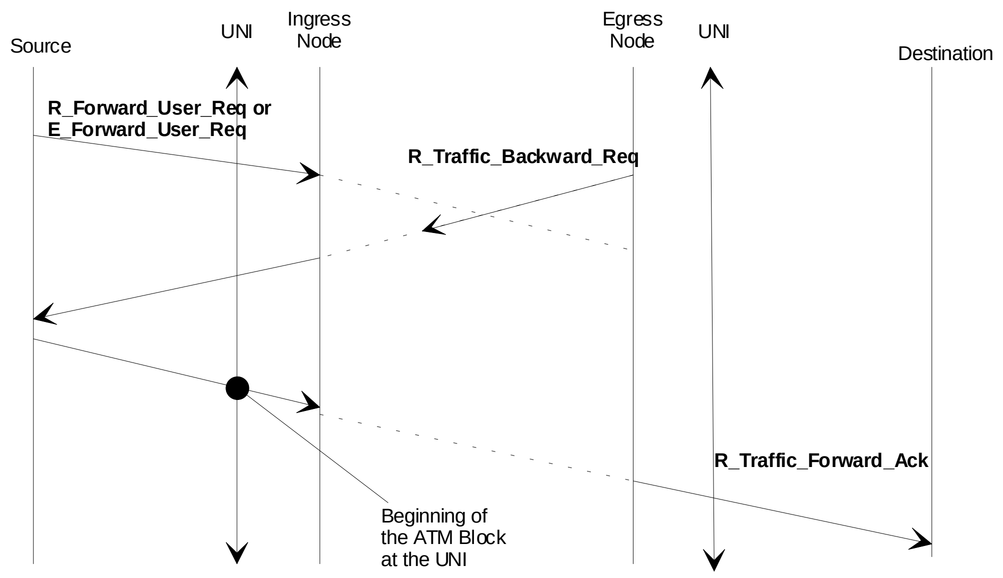

Figure 21: Backward Traffic Control procedure in the rigid mode with collision with a Forward BCR negotiation procedure

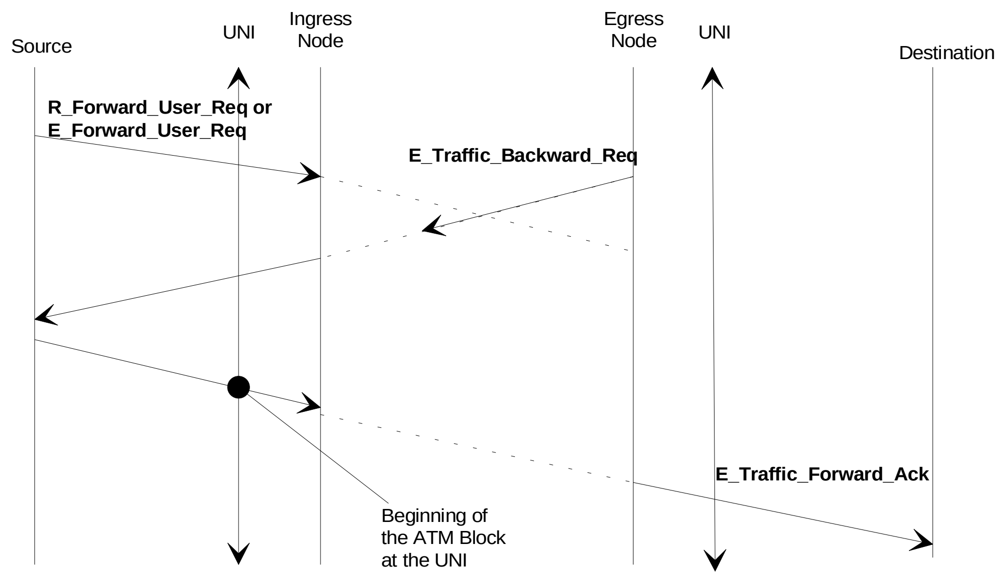

Figure 22: Backward Traffic Control procedure in the elastic mode with collision with a Forward BCR negotiation procedure

5.7.3.4 Collision between forward and backward traffic control procedures

(In replacement of editor's note). When a Forward Traffic Control collides with a Backward Traffic Control procedure the former takes priority and is completed in the normal way (see subclauses 5.6.3.1.1 and 5.6.3.2.1) whereas the Backward Traffic Control is aborted. Upon reception at a network node of a Forward Traffic Control Request message, any BCR reservation made by the Backward Traffic Control is cancelled. On the other hand any Backward Traffic Control message arriving at a network node while some resources have been reserved by a Forward Traffic Control message is ignored by the network node and the M.CI field is set, if necessary. No acknowledgement to this request is sent to the source. The messages exchanged across the UNI are shown in figures 23 and 24 in the case of Rigid and Elastic Forward Traffic Control, respectively.

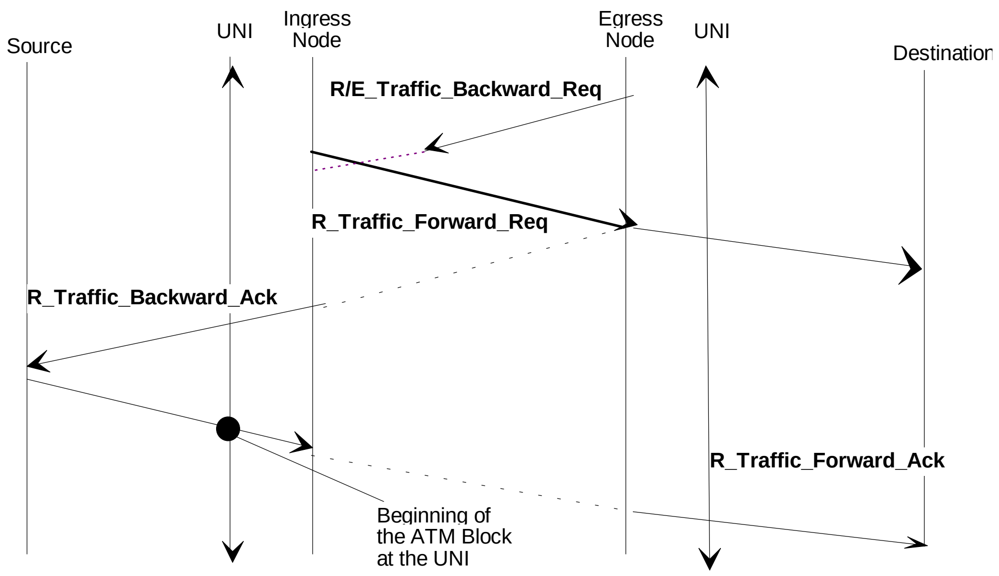

Figure 23: Forward Traffic Control procedure in the rigid mode with collision with a Backward Traffic Control negotiation procedure

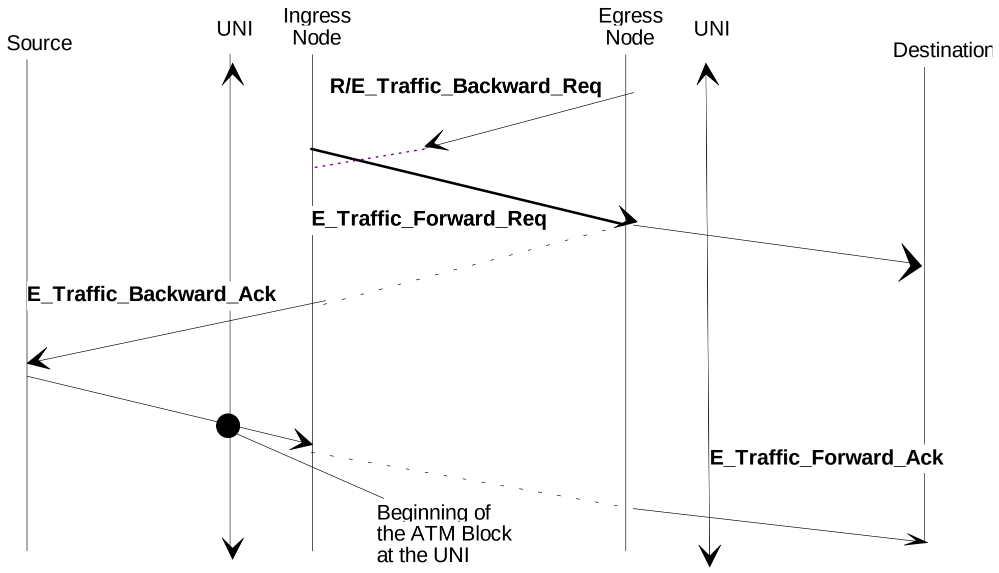

Figure 24: Forward Traffic Control procedure in the elastic mode with collision with a Backward Traffic Control negotiation procedure

6 Cases of possible applications

6.1 Support of A-VPN

A possible application of the ABT-DT protocol described in the previous subclauses is the support of Broadband Virtual Private Networks (B-VPN), specially in the context of a semi-permanent ATM environment. In the following this application will be addressed by the term "A-VPN".

In this application, the idea of disassociating connectivity from bandwidth allocation is still a basic concept: the customer leases from the public network a mesh of DBR VP "links", which topologically connects the various sites of its private network across the public one; in addiction, a full mesh of VCCs connects each pair of locations belonging to the AVPN, but no resources are associated to them, the VCCs being only a logical concatenation of VP links. The total bandwidth of the VP link is shared by the VCCs, and it is even possible for a single VCC to consume the total bandwidth of one VP link. The concept of the A-VPN is captured by figure 25.

If no controls are applied to the VCCs overload on a "link" might arise every time two or more VCCs which share some "link", currently require a greater bandwidth than that of the VPC "link". Moreover, as inside the ATM nodes the A-VPN traffic, in general, shares resources with the traffic of other customers, the degradation of performance due to overload also affects the QoS of connections not belonging to the A-VPN responsible for it.

In the above scenario overload occurs due to "structural" unawareness of the ATM network about the behaviour of the A-VPN users and the current load of the VCCs. VCCs cannot be policed by ATM, as the only traffic contract between ATM operator and A-VPN user refers to VPC "links", while the VCCs are completely under the user control only; any VCC can, in principle, reach the total capacity of the smaller VPC "link" connecting one of the end point of the VCC to the first ATM node; so ATM network results practically unprotected.

The above overload problems can efficiently be solved , by the utilization of ABT-DT VCCs: the ABT-DT protocol, indeed, will allow the interworking between the "private" network management, responsible for the operation of the A-VPN, and the public network management, which should guarantee network performance objective both for the private A-VPN and for the rest of the services contemporarily supported over the public ATM network, avoiding undue interference between the traffic of one service and the others.

The ABT-DT protocol, in fact, provides a way to control the actual allocation of resources to the various VCCs, avoiding inconsistency which can overload some of the VP links and some of the nodes. Before an ABT-DT VCC can use a certain amount of resources, it should ask the public ATM network for its availability in the VP links involved. The bandwidth of the VCC can only be enlarged if the answer is positive. Resources remain allocated to a VCC until it decides to release them, for use by other VCCs belonging to the AVPN.

By means of the ABT-DT protocol the network keeps track of the current usage of the resources of the VP links, and is allowed to deny VCC usage if some of the VP links become overloaded. Furthermore, the network is aware of the amount of bandwidth allocated to each VCC and can enforce it at the network ingress, provided that the parameters of the UPC mechanism can be dynamically changed according to the results of the bandwidth negotiation procedures.

In this approach, resources of the VP links are dynamically allocated and released by its VCC, the requests being driven by the traffic fluctuations at the VCC level.

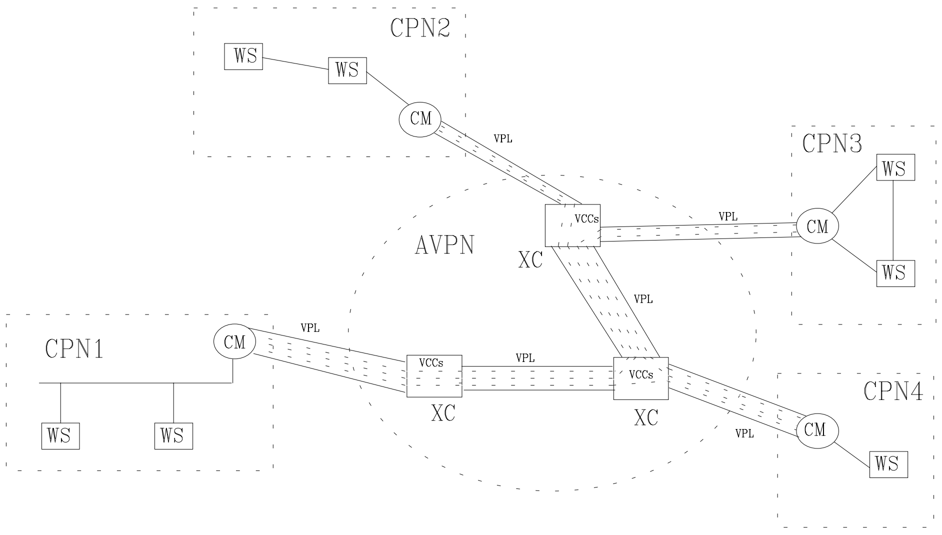

Figure 25: A-VPN concept

It is the user’s responsibility, where the word "user" refers to the management of the AVPN, to correctly dimension his private network in terms of number, topology and bandwidth of the VP links, and to define the routeing of the VCC among the different CNPs through the AVPN links. The public network should be provided with units able to run the ABT-DT protocol.

It is to be noted that, since each AVPN is supported over dedicated VPCs, the segregation of the AVPN traffic only requires that the resources allocated to these VPCs are not exceeded and this result can be achieved with the basic BCR negotiation procedures provided by the ABT-DT protocol; therefore, in order to support AVPN, a simplified version of the ABT-DT protocol could be sufficient. In particular, since the competition to access the available resources is limited to the VCCs belonging to the same customer, the VCCs do not need to have any guaranteed QoS at the block level so that the Traffic Control procedures as well as the Network Maintenance procedures foreseen by the protocol do not need to be implemented.

Annex A: Reference behaviours

A.1 Source reference behaviour

Any BCR modification by the source is initiated by sending a Bandwidth Request RM cell on the forward direction with DIR=0 and the requested BCR. The BCR request RM cells (forward or backward requests) generated by the source should be conforming to GCRA(T_{RM},τ_{RM}); any non conforming RM cell may be ignored by the network. Moreover, the source should not initiate a BCR modification in the forward direction while another one is still in progress within the network (i.e., the source has not received any response from the network); otherwise, the source receives a specific maintenance message (Not_Ready message) from the network.

For a given allocated BCR value λ for the CLP=0+1 cell flow, the source should deliver a cell flow conforming to the allocated BCR λ and the corresponding CDV tolerance $\tau(\lambda)$. The OAM cell flow should be conforming to the allocated BCR value and the associated CDV tolerance satisfying the general requirements given in Appendix II to ETS 300 301.

The source may receive:

1) a Bandwidth Acknowledgement RM cell with Traffic Management=0 corresponding to a BCR request initiated by the source. If CI=0 or Elastic=0, the source should acknowledge this message by sending in the forward direction a Bandwidth Acknowledgement RM cell with BCR values less than or equal to those communicated by the network;

2) a Bandwidth Request RM cell with Traffic Management=0/1, indicating a BCR negotiation initiated by the destination, a traffic control procedure at network egress, or the loss of some RM cells. This message may be received even if the source has not initiated any BCR negotiation. If Traffic Management=1 or (Traffic Management=0 and (CI=0 or Elastic=0)), this message should be acknowledged by sending a Bandwidth Acknowledgement RM cell with BCR values as above;

3) a Bandwidth Acknowledgement RM cell with Traffic Management=1 indicating a traffic control procedure at network ingress or the loss of some RM cells. This message may be received even if the source has not initiated any BCR negotiation. The source should acknowledge this message by sending in the forward direction a Bandwidth Acknowledgement RM cell with correct BCR values.

The state of the source is characterized by two values IDLE and HUNT. The source is in state IDLE when no BCR negotiation is in progress. The source is in state HUNT when a BCR negotiation has been initiated by the source. The source should get a response from the network within a time interval at most W_Source time unit long.

The pseudo code of the source behaviour is given in table A.1. The source process is related to the ingress process via the DoS (downstream) and UpS (Upstream) routes. The following notation is used in the SDL diagrams:

- T stands for true and F for false in Boolean expressions;
- binary flags are coded by Boolean instead of 0 and 1. Specifically 1 is coded by T except for the Elastic bit where it is coded by F.

Messages are represented by two SDL signals:

1) FMsg(kind,bandwidth, ci, el, mt, trf, seq) for messages with the Dir bit set to 0;
2) BMsg(kind,bandwidth, ci, el, mt, trf, seq) for messages with the Dir bit set to 1.

where:

- kind: message type (Ack or Req);
- rate : vector of BCR values for the CLP=0+1 and OAM cell flows;
- ci : congestion indication (Boolean);
- el : elasticity indication (Boolean);
- mt: maintenance indication (Boolean);
- trf: traffic management indication (Boolean);

- seq: sequence number (integer).

The BCR currently allocated at the source to the different cell flows of the connection are denoted by $\Lambda_{Source}(0+1)$ and $\Lambda_{Source}(OAM)$. The BCR acknowledged by the source should be less than or equal to the BCR communicated by the network and are computed by the procedure **adapt**(M) as a function of the message received from the network. Moreover, the following procedures are defined:

- the **check_number**(M) procedure checks whether a traffic message M has a correct sequence number with respect to the value of the seq_number variable; the result is either true or false;

- the **change_number**(M) procedure updates the sequence number by incrementing its value by 1 modulo 2^{32}.

Table A.1: Pseudo code of the source behaviour

```
PROCESS Source;
 SYNONYM $T_{Source}$ duration = EXTERNAL;

 DCL $\Lambda_{Source}$,Λ Bandwidth;
 DCL seq,sn Integer;
 DCL kind Kind;
 DCL ci,ci0,el,el0,mt0,mt,tf Boolean;

 TIMER $W_{Source}$;
 PROCEDURE check_number REFERENCED;
 PROCEDURE Change_number REFERENCED;
 PROCEDURE Adapt REFERENCED;

 START ;
  TASK sn:=0;
  NEXTSTATE Idle;

 STATE Idle;

  INPUT Change_BCR(Λ);
   COMMENT 'Rigid BCR negotiation';
   TASK el := F, mt := F;
Modify:
   CALL Change_number;
   OUTPUT FMsg(Req,Λ,F,el,mt,F,sn) VIA UpS;
   DECISION Λ <= $\Lambda_{Source}$;
   (T):
    CALL adapt;
    NEXTSTATE Idle;
   (F):
    SET(NOW+$T_{Source}$,$W_{Source}$);
    NEXTSTATE Hunt;
   ENDDECISION;

  INPUT Change_BCR_EL(Λ);
   COMMENT 'Elastic BCR negotiation';
   TASK el := T, mt := F;
   JOIN Modify;

  INPUT Status_Enquiry;
   COMMENT 'status enquiry';
   TASK Λ := Null, el := T, mt := T;
   CALL Change_number;
   OUTPUT FMsg(Req,Λ,F,el,mt,F,sn) VIA UpS;
   SET(NOW+$T_{Source}$,$W_{Source}$);
   NEXTSTATE Hunt;

  INPUT BMsg(kind,Λ,ci,el0,mt0,tf,seq);
   DECISION kind;
   (Req):
    JOIN SrcReq;
   (Ack):
    DECISION tf;
    (F):
    (T):
     JOIN SrcTrf;
    ENDDECISION;
   ENDDECISION;
   NEXTSTATE Idle;

 STATE Hunt;

  SAVE Change_BCR;
  SAVE Change_BCR_EL;
  SAVE Status_Enquiry;

  INPUT BMsg(kind,Λ,ci,el0,mt0,tf,seq);
   DECISION kind;
    (Req):
     RESET($W_{Source}$);
SrcReq:
     COMMENT 'Request from the egress or the
               destination';
     DECISION tf;
     (F):
      COMMENT 'User BCR negotiation';
      DECISION ci and not(el0);
      (T):
       COMMENT 'that failed';
      (F):
       COMMENT 'that succeeded';
       CALL Adapt;
       OUTPUT FMsg(Ack,Λ,ci,el0,mt0,tf,seq) VIA UpS;
      ENDDECISION;
     (T):
      JOIN SrcTrf;
     ENDDECISION;
    (Ack):
     COMMENT 'Treatment of acknowledgements';
     DECISION tf;
     (F):
      DECISION mt0 and ci;
      (T):
       NEXTSTATE -;
      (F):
       COMMENT 'Checks the sequence number before
           reseting the timer';
       CALL check_number(ci0);
       DECISION ci0;
       (T):
        RESET($W_{Source}$);
       (F):
        NEXTSTATE -;
       ENDDECISION;
       COMMENT 'In any case, adapt to its contents';
       DECISION not(ci) or el0;
       (T):
        CALL Adapt;
        OUTPUT FMsg(Ack,Λ,ci,el0,mt0,tf,seq) VIA UpS;
       (F):
       ENDDECISION;
      ENDDECISION;
     (T):
      RESET($W_{Source}$);
SrcTrf:
      COMMENT 'Traffic messages (Req or Ack) are
          accepted as is';
      CALL Adapt;
      OUTPUT FMsg(Ack,Λ,ci,el0,mt0,tf,seq) VIA UpS;
     ENDDECISION;
    ENDDECISION;
    NEXTSTATE Idle;

   INPUT $W_{Source}$;
    COMMENT 'Retry on failure';
    CALL Change_number;
    SET(NOW+$T_{Source}$,$W_{Source}$);
    OUTPUT FMsg(Req,Λ,F,el,mt,F,sn) VIA UpS;
    NEXTSTATE -;
ENDPROCESSSource
```

A.2 Destination reference behaviour

The destination may receive messages from the network due to BCR negotiations initiated by the source or the network (traffic control procedures). Moreover, the destination may initiate a BCR negotiation applicable to the traffic generated by the source. For this purpose, the destination should send a Bandwidth Request RM cell with DIR=1 and the requested BCR values. In response to this request, the destination may receive:

1) a Bandwidth Acknowledgement RM cell with Traffic Management=0, in response to the BCR request initiated by the destination;

2) a Bandwidth Request RM cell with Traffic Management=1 and Maintenance=1 due to a traffic control procedure at network ingress (RM cell loss);

3) a Bandwidth Acknowledgement RM cell with Traffic Management=1 and Maintenance=0 due to a traffic control procedure at network egress; or

4) a Bandwidth Acknowledgement RM cell with Traffic Management=1 and Maintenance=1 due to the loss of some RM cells.

The destination has to accept a BCR negotiation. When the destination submits BCR request RM cells (forward or backward requests) to the network, they have to be conforming to the GCRA(T'_{RM},τ'_{RM}); any non conforming RM cell may be ignored by the network.

The pseudo code of the destination is given in table A.2. The destination process is related to the network via the DoS (downstream) and UpS (upstream) routes. The following procedures are defined:

- the **check_number**(M) procedure checks whether a traffic message M has a correct sequence number with respect to the value of the seq_number variable; the result is either true or false;

- the **change_number**(M) procedure updates the sequence number by incrementing its value by 1 modulo 2^{32}.

Table A.2: Pseudo code of the destination behaviour

```
PROCESS Destination;
  DCL $\Lambda$,$\Lambda_{Destination}$ Bandwidth;
  DCL sn,seq Integer;
  DCL kind Kind;
  DCL ci,ci0,el,el0,mt,mt0,tf Boolean;
  DCL $T_{Dest}$ duration;
  TIMER $W_{Dest}$;

  PROCEDURE check_number REFERENCED;
  PROCEDURE Change_number REFERENCED;

  START ;
    TASK sn := 1;
    TASK $T_{Dest}$ := $T_{usr}$;
    NEXTSTATE Idle;

  STATE Idle;

    INPUT Change_BCR($\Lambda$);
      COMMENT 'Rigid BCR negotiation';
      TASK el := F, mt := F;
      CALL Change_number;
      OUTPUT BMsg(Req,$\Lambda$,F,el,mt,F,sn) VIA DoS;
      SET(NOW+$T_{Dest}$,$W_{Dest}$);
      NEXTSTATE Hunt;

    INPUT Change_BCR_EL($\Lambda$);
      COMMENT 'Elastic BCR negotiation';
      TASK el := T, mt := F;
      CALL Change_number;
      OUTPUT BMsg(Req,$\Lambda$,F,el,mt,F,sn) VIA DoS;
      SET(NOW+$T_{Dest}$,$W_{Dest}$);
      NEXTSTATE Hunt;

    INPUT Status_Enquiry;
      COMMENT 'status enquiry';
      TASK $\Lambda$ := Null, el := T, mt := T;
      CALL Change_number;
      OUTPUT BMsg(Req,$\Lambda$,F,el,mt,F,sn) VIA DoS;
      SET(NOW+$T_{Dest}$,$W_{Dest}$);
      NEXTSTATE Hunt;

    INPUT FMsg(kind,$\Lambda$,ci,el0,mt0,tf,seq);
      DECISION kind;
      (Req):
        DestReqForward:
        DECISION not(el0) and $\Lambda <= \Lambda_{Destination}$ and
                 not(tf) and not(mt0);
        (T):
          COMMENT 'BCR decrease';
          TASK $\Lambda_{Destination}$ := $\Lambda$;
          NEXTSTATE Idle;
        (F):
          COMMENT 'other request: wait for
acknowledgement';
          SET(NOW+$T_{Dest}$,$W_{Dest}$);
          NEXTSTATE Waiting;
        ENDDECISION;
      (Ack):
        DECISION tf;
        (T):
          COMMENT 'Only kind of Ack that may arrive
                   (traffic from egress)';
          TASK $\Lambda_{Destination}$ := $\Lambda$;
        (F):
        ENDDECISION;
      ENDDECISION;
      NEXTSTATE -;

  STATE Waiting;

    SAVE Change_BCR;
    SAVE Change_BCR_el;
    SAVE Status_Enquiry;

    INPUT  FMsg(kind,$\Lambda$,ci,el0,mt0,tf,seq);
      COMMENT 'Blocked until an Ack arrives';
      DECISION kind;
      (Req):
        NEXTSTATE -;
      (Ack):
        RESET($W_{Dest}$);
        TASK $\Lambda_{Destination}$ := $\Lambda$;
        NEXTSTATE Idle;
      ENDDECISION;

    INPUT $W_{Dest}$;
      COMMENT 'Timeout - back to normal state';
      NEXTSTATE Idle;

  STATE Hunt;

    SAVE Change_BCR;
    SAVE Change_BCR_el;
    SAVE Status_Enquiry;

    INPUT $W_{Dest}$;
      COMMENT 'Timeout - retries the negotiation';
      CALL Change_number;
      SET(NOW+$T_{Dest}$,$W_{Dest}$);
      OUTPUT BMsg(Req,$\Lambda$,F,el,mt,F,sn) VIA DoS;
      NEXTSTATE -;

    INPUT FMsg(kind,$\Lambda$,ci,el0,mt0,tf,seq);
      DECISION kind;
      (Req):
        RESET($W_{Dest}$);
        JOINT DestReqForward;
      (Ack):
        DECISION mt0 and ci;
        (T):
          COMMENT 'Not ready';
        (F):
          COMMENT 'Updates according to the
Acknowledge';
          TASK $\Lambda_{Destination}$ := $\Lambda$;
          CALL check_number(ci0);
          DECISION ci0 or tf;
          (T):
            RESET($W_{Dest}$);
            NEXTSTATE Idle;
          (F):
          ENDDECISION;
        ENDDECISION;
        NEXTSTATE -;
      ENDDECISION;
ENDPROCESSDestination
```

A.3 Network element reference behaviour

The ingress and egress network nodes should perform some UPC/NPC functions. For instance, they may control RM cell flows by discarding other RM cells than ABT/DT RM cells and by enforcing the peak cell rate of any ABT/DT bandwidth request RM cell flow, by means of the GCRA(T_{RM},τ_{RM}) and GCRA(T'_{RM},τ'_{RM}). They should also process the conforming ABT/DT cells to run the ABT/DT procedure. Specifically, special functions located in the ingress and egress nodes should be able to guarantee the correct exchange of ABT/DT cells. For this purpose, such a function should be able to handle ABT/DT RM cells propagating on the forward as well as backward directions.

A.3.1 Ingress node

An ingress node is modelled by a four state process:

- NORMAL: there is no BCR negotiation in progress;
- TRAFFIC: a traffic control procedure has been initiated and is still in progress;
- ACKNOWLEDGING: an RM cell has been sent to the source and the process is waiting for a response.

WAITING: some resource are booked and the process is waiting for an acknowledgement RM cell.

A node maintain two couples of BCR values, namely:

- the BCR Λ currently allocated to the cell flows at network ingress;
- the BCR Λ’ reserved for the cell flows at network ingress.

The book(M,U) procedure updates the reserved and allocated BCR values maintained in an ABT/DT unit. For instance the **book**(M,U) procedure at network ingress can be coded as:

```
PROCEDURE book;
FPAR M,U : bandwidth;
  START
    TASK Λ (0+1) := M(0+1);
    TASK Λ(OAM) := min(M(0+1),M(OAM));
    TASK Λ’(0+1) := U(0+1);
    TASK Λ’(OAM) := min(U(0+1),U(OAM));
    RETUN
ENDPROCEDURE
```

To handle traffic control message, an implicit variable seq_number is defined. It stores the sequence number of the last relevant traffic control message. With regard to the sequence number, the following procedures are defined:

- the **check_number**(M) procedure checks whether a traffic message M has a correct sequence number with respect to the value of the seq_number variable; the result is either true or false;
- the **change_number**(M) procedure updates the sequence number by incrementing its value by 1 modulo 2^{32};
- the **attribute_msg**(M) procedure assigns the value of the seq_number to the sequence number of message M.

The ingress ABT/DT process is linked to the source process by the UpS (upstream) route and to the first intermediate process by the DoS (downstream) route. The pseudo code of the ingress node is given in table A.3.

Table A.3: Pseudo code of the ingress node

```
PROCESS Ingress;

 SYNONYM Imax Bandwidth = EXTERNAL;
 SYNONYM Ipcr Bandwidth = EXTERNAL;
 SYNONYM $T_{ingress}$ Duration = EXTERNAL;

 DCL $\Lambda$,$\Lambda'$,rate Bandwidth;
 DCL seq,seq_counter,ignored_seq Integer;
 DCL kind Kind;
 DCL ci,el,mt,tf,auxiliary,el_memory Boolean;
 TIMER $W_{ingress}$;

 PROCEDURE Book REFERENCED;
 PROCEDURE change_number REFERENCED;
 PROCEDURE attribute_msg REFERENCED;
 PROCEDURE check_number REFERENCED;

 START;

  TASK $\Lambda$ := Null, $\Lambda'$ := Null;
  NEXTSTATE Normal;

 STATE Normal;

  INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req) :
    DECISION tf;
    (T):
     COMMENT 'Traffic request';
     COMMENT 'On a UNI node : error from the source';
     TASK tf := F;
     JOIN IngressNormalForward;
    (F):
     COMMENT 'User request';
     IngressNormalForward:
     TASK rate := min(Imax,rate);
     COMMENT 'Status enquiry or BCR increase';
     DECISION (mt or rate > $\Lambda$);
     (T):
      DECISION mt;
      (T):
       TASK rate := $\Lambda$;
      (F):
      ENDDECISION; /* mt */
      CALL Book($\Lambda$,rate);
      TASK el_memory := el;
      SET (NOW+$T_{ingress}$, $W_{ingress}$);
      OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
      NEXTSTATE Waiting;
     (F):
      COMMENT 'BCR decreased (not acknowledged)';
      TASK el := F;
      CALL Book(rate,Null);
      OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
      NEXTSTATE Normal;
     ENDDECISION; /* mt or rate \(>\) $\Lambda$ */
    ENDDECISION; /* tf */
   (Ack) :
    COMMENT 'No ACK in this state';
    NEXTSTATE Normal;
   ENDDECISION; /* kind */

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req) :
    IngressBackward:
    DECISION tf;
    (T):
     COMMENT 'Traffic message from the egress';
     IngressTrafficBackward:
      CALL Book($\Lambda$,rate);
      OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
      NEXTSTATE Acknowledging;
    (F):
     COMMENT 'BCR negotiation from the destination';
     DECISION el or not(ci);
     (T):
      COMMENT 'successful';
      CALL Book($\Lambda$,rate);
      OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
     (F):
      COMMENT 'failed';
      CALL Book($\Lambda$,Null);
      OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
      COMMENT 'On the UNI, it is looped back
           to the destination';
       TASK rate := $\Lambda$;
       TASK kind := Ack;
       OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
       NEXTSTATE Normal;
     ENDDECISION; /* el or mt */
    ENDDECISION; /* tf */
    NEXTSTATE Acknowledging;
   (Ack):
    NEXTSTATE -;
   ENDDECISION; /* kind */

  INPUT POL(el);
   COMMENT 'Policing action from the ingress';
   IngressPolicing:
   DECISION el;
   (T):
    TASK rate := Ipcr;
   (F):
    TASK rate := $\Lambda$;
   ENDDECISION; /* el */
   TASK el_memory:=el;
   SET(NOW+$T_{ingress}$,$W_{ingress}$);
   TASK kind := Req, mt := F, tf := T, ci := F;
   CALL Book($\Lambda$,rate);
   CALL change_number;
   CALL attribute_msg;
   OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
   NEXTSTATE Traffic;

 STATE Acknowledging;

  SAVE POL;

  INPUT FMsg(kind,rate,ci,el,mt,tf,ignored_seq);
   DECISION kind;
   (Req) :
    NotReadyIngress:
    COMMENT 'A Not Ready is sent';
    OUTPUT BMsg(Ack,rate,T,T,T,F,0) VIA DoS;
    NEXTSTATE -;

   (Ack) :
    COMMENT 'The expected acknowledgement from the
         source';
    TASK rate := min(rate,$\Lambda'$);
    CALL Book(rate,Null);
    OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
    NEXTSTATE Normal;
   ENDDECISION; /* kind */

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
   COMMENT 'Only traffic messages from the egress
        are considered';
    DECISION kind;
    (Req) :
      DECISION tf;
      (T):
        JOIN IngressTrafficBackward;
      (F):
      ENDDECISION;
    (Ack) :
    ENDDECISION;
    NEXTSTATE -;

STATE Waiting;

  INPUT POL(el);
    COMMENT 'Same as in state Normal';
    JOIN IngressPolicing;

  INPUT $W_{ingress}$;
    COMMENT 'The timer has expired, an error recovery
            message is sent';
    TimerIngress:
    SET(NOW+$T_{ingress}$,$W_{ingress}$);
    CALL change_number;
    CALL attribute_msg;
    OUTPUT FMsg(Req,Λ',F,el_memory,T,T,seq) VIA UpS;
    NEXTSTATE Traffic;

  INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
    JOIN NotReadyIngress;

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
    RESET($W_{ingress}$);
    DECISION kind;
    (Req) :
      COMMENT 'Requests from the destination or egress
              have priority over requests from the source';
      DECISION tf;
      (T):
        JOIN IngressTrafficBackward;
      (F):
        DECISION el or not(ci);
        (T):
          CALL Book(Λ,rate);
          OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
        (F):
          COMMENT 'On failure always wait for an acknow-
                  ledgement from the source to show that its
                  negotiation has failed';
          CALL Book(Λ,Λ);
          OUTPUT BMsg(kind,Λ,ci,el,mt,tf,seq) VIA DoS;
        ENDDECISION;
        NEXTSTATE Acknowledging;
      ENDDECISION;
    (Ack) :
      COMMENT 'Acknowledgement to the source BCR
              modification';
      DECISION not(ci) or el;
      (T):
        COMMENT 'Successful negotiation';
        CALL Book(Λ,rate);
        OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
        NEXTSTATE Acknowledging;
      (F):
        COMMENT 'Unsuccessful negotiation
                (No ack expected)';
        CALL Book(Λ,Null);
        OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
        NEXTSTATE Normal;
      ENDDECISION; /* not(ci) or el */
    ENDDECISION; /* kind */

 STATE Traffic;

  SAVE Pol;

  INPUT $W_{ingress}$;
    JOIN TimerIngress;

  INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
    JOIN NotReadyIngress;

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
    DECISION tf and (kind=Ack);
    (T):
      COMMENT 'Acknowledgement to a traffic message';
      CALL check_number(auxiliary);
      DECISION auxiliary;
      (T):
        RESET($W_{ingress}$);
          DECISION ci or (el and not(mt));
          (T):
            CALL Book(Λ,rate);
            OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
            NEXTSTATE Acknowledging;
          (F):
            COMMENT 'if no modification, directly
                    answer without acknowledgement
                    from the source';
            CALL Book(Λ,Null);
            TASK rate := Λ;
            OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
            OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
            NEXTSTATE Normal;
          ENDDECISION; /* ci or (el and not(mt)) */
      (F):
        COMMENT 'old message with wrong sequence
number';
      ENDDECISION; /* auxiliary */
    (F):
      COMMENT 'Other messages are ignored';
    ENDDECISION;  /* tf and (kind=Ack) */
    NEXTSTATE -;

ENDPROCESSIngress;
```

A.3.2 Intermediate node

An intermediate node process has three states :

- NORMAL: no BCR negotiation is in progress;
- WAITING: a user negotiation has been initiated and is still in progress;
- TRAFFIC: a traffic control procedure has been initiated and is still in progress.

The **possible** and **fairshare** procedures that control allocation respectively in the rigid and elastic case are network operator specific.

As for the ingress process, two variables Λ and Λ' are used. Note that the **book** procedure should also modify the available bandwidth Δ in a network node. Moreover, information on OAM traffic is irrelevant. As a consequence the **book** procedure can be coded as :

```
PROCEDURE book;
  FPAR M,U : bandwidth;
  START;
    TASK $\Lambda$ := M(0+1);
    TASK $\Lambda'$ := U(0+1);
    TASK $\Delta$ := $\Delta$ + max ($\Lambda$(0+1),$\Lambda'$(0+1)) - max(M(0+1),U(0+1));
    RETURN;
ENDPROCEDURE;
```

As an example, if the BCR negotiation context is not taken into account in the allocation of network resources, the **possible** procedure may be defined as follows:

```
PROCEDURE possible;
  FPAR IN $\lambda$ bandwidth;
           IN mt Boolean;
           IN/OUT $\lambda_r$ bandwidth;
  START;
    TASK $\lambda_r$ := min($\lambda$,$\Delta$+max($\lambda$,$\Lambda'$));
    RETURN;
ENDPROCEDURE;
```

Under the same assumptions and in the case of unweighted fairness, the **fairshare** procedure may be defined as :

```
PROCEDURE fairshare;
  FPAR $\lambda$ bandwidth;
           IN/OUT $\lambda_r$ bandwidth;
           IN/OUT ci bandwidth;
  START;
     TASK $\lambda_r$ := min(C/N, $\Delta$+$\Lambda$);
     TASK $\lambda_r$ := min($\lambda_r$,$\lambda$(0+1));
     TASK ci := ($\lambda_r$ < C/N);
     RETURN;
ENDPROCEDURE;
```

where N is the number of ABT/DT connexions and C is the total amount of bandwidth for ABT/DT connections.

The pseudo code of an intermediate node is given in table A.4.

Table A.4: Pseudo code of an intermediate node

```
PROCESS PInt;
 DCL Λ,Λ0,Λ',avlb,rate Bandwidth;
 DCL seq Integer;
 DCL kind Kind;
 DCL ci,ci0,el,mt,tf Boolean;

 PROCEDURE Book REFERENCED;
 PROCEDURE possible REFERENCED;
 PROCEDURE Fairshare REFERENCED;
 PROCEDURE Forgetting REFERENCED;

 START;

  TASK Λ := Null, Λ' := Null;
  NEXTSTATE Normal;

 STATE Normal;

  INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req):
    IntFwdTraffic:
    DECISION tf;
    (T):
     COMMENT 'Traffic request from the ingress';
     DECISION el;
     (T):
      COMMENT 'with elastic parameters';
      CALL Fairshare(rate,Λ0,ci0);
      TASK ci := ci or ci0, rate := Λ0;
     (F):
      COMMENT 'with rigid parameters';
      CALL possible(rate,mt,Λ0);
      DECISION rate > Λ0;
      (T):
       TASK ci := T, rate := Λ0;
      (F):
      ENDDECISION;
     ENDDECISION; /* el */
     CALL Book(Λ,rate);
     OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
     NEXTSTATE Traffic;
    (F):
     COMMENT 'Request from the source';
     DECISION mt;
     (T):
      COMMENT 'Status enquiry';
      CALL possible(rate,T,Λ0);
      TASK rate := min(rate,Λ0);
      CALL Book(Λ,rate);
     (F):
      DECISION el;
      (T):
       COMMENT 'Elastic request';
       CALL Fairshare(rate,Λ0,ci0);
       TASK ci := ci or ci0, rate := Λ0;
       CALL Book(Λ,rate);
      (F):
       DECISION ci;
       (F):
        DECISION (rate <= Λ);
        (T):
         COMMENT 'BCR decrease';
         CALL Book(rate, Null);
         OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
         NEXTSTATE Normal;
        (F):
         COMMENT 'Rigid BCR increase ...';
         CALL possible(rate,mt,Λ0);
         TASK ci := IF rate > Λ0 THEN T ELSE ci FI;
         DECISION ci;
         (F):
          COMMENT '... that succeeds';
          CALL Book(Λ,rate);
         (T):
          COMMENT '... that fails';
         ENDDECISION;
        ENDDECISION; /* rate ≤ Λ */
       (T):
        COMMENT 'request that has already failed';
       ENDDECISION; /* ci */
      ENDDECISION; /* el */
     ENDDECISION; /* mt */
     COMMENT 'in any case, the message is forwarded';
     OUTPUT FMsg(Req,rate,ci,el,mt,tf,seq) VIA UpS;
     NEXTSTATE Waiting;
    ENDDECISION; /* tf */
   (Ack):
    COMMENT 'There should be no acknowledgement
         in this state';
    NEXTSTATE Normal;
   ENDDECISION;

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
   IntBwd:
   DECISION kind;
   (Req):
    DECISION tf;
    (T):
     COMMENT 'Traffic request from the egress';
     DECISION el;
     (T):
      COMMENT 'with elastic parameters';
      CALL Fairshare(rate,Λ0,ci0);
      TASK ci := ci or ci0, rate := Λ0;
     (F):
      COMMENT 'with rigid parameters';
      CALL possible(rate,mt,Λ0);
      DECISION rate > Λ0;
      (T):
       TASK ci := T, rate := Λ0;
      (F):
      ENDDECISION;
     ENDDECISION;
     CALL Book(Λ,rate);
     OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
     NEXTSTATE Traffic;
    (F):
     COMMENT 'Request from the destination';
     DECISION mt;
     (T):
      COMMENT 'status enquiry';
      CALL possible(rate,T,Λ0);
      TASK rate := min(rate,Λ0);
      CALL Book(Λ,rate);
     (F):
      DECISION el;
      (T):
       COMMENT 'elastic BCR modification';
       CALL Fairshare(rate,Λ0,ci0);
       TASK ci := ci or ci0, rate := Λ0;
       CALL Book(Λ,rate);
      (F):
       DECISION ci;
       (F):
        COMMENT 'rigid BCR modification';
        CALL possible(rate,mt,Λ0);
        TASK ci := IF rate > Λ0 THEN T ELSE ci FI;
        DECISION ci0;
        (F):
          CALL Book(Λ,rate);
         (T):
          TASK ci := T;
         ENDDECISION;
        (T):
         COMMENT 'failed request';
        ENDDECISION;
       ENDDECISION;
      ENDDECISION;
      OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
      NEXTSTATE Waiting;

     ENDDECISION;
    (Ack):
     COMMENT 'No acknowledgement in this state';
     NEXTSTATE Normal;
    ENDDECISION;

STATE Waiting;

  INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req):
    DECISION tf;
    (T):
     COMMENT 'Traffic request have priority';
     JOIN IntFwdTraffic;
    (F):
     COMMENT 'Request from the source are ignored
              (collision principle)';
     TASK ci := T;
     OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
     NEXTSTATE Waiting;
    ENDDECISION;
   (Ack):
    COMMENT 'Acknowledgement from the source is taken
             into account';
    CALL Book(rate, Null);
    OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
    NEXTSTATE Normal;
   ENDDECISION;

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req):
    JOIN IntBwd;
   (Ack):
    DECISION ci and not(el) and not(mt) and not(tf);
    (T):
     COMMENT 'rigid source BCR increase has failed,
              everything is cleared';
     CALL Book(Λ, Null);
     OUTPUT BMsg(Ack,rate,ci,el,mt,tf,seq) VIA DoS;
     NEXTSTATE Normal;
    (F):
     DECISION el;
     (T):
      COMMENT 'if the negotiation was elastic, just keeps
               what can actually be used';
      CALL Book(Λ,rate);
     (F):
      COMMENT 'otherwise just pass it to the source';
     ENDDECISION;
    ENDDECISION;
    OUTPUT BMsg(Ack,rate,ci,el,mt,tf,seq) VIA DoS;
    NEXTSTATE Waiting;
   ENDDECISION;

STATE Traffic;

  INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req):
    DECISION tf;
    (T):
     COMMENT 'Traffic message from the source have the
              highest precedence';
     JOIN IntFwdTraffic;
    (F):
     COMMENT 'Other messages are ignored';
     OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
     NEXTSTATE -;
    ENDDECISION;
   (Ack):
    DECISION tf;
    (T):
     COMMENT 'The acknowledgement is taken
              into account';
     CALL Book(rate,Null);
     OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
     NEXTSTATE Normal;
    (F):
     COMMENT 'Not the acknowledgement expected';
     OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
     NEXTSTATE -;
    ENDDECISION;
   ENDDECISION;

  INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
   DECISION kind;
   (Req):
    DECISION tf and mt;
    (T):
     COMMENT 'error recovery from a traffic message
              emited by the egress : nothing is booked but
              the computation is done';
     CALL possible(rate,mt,Λ0);
     DECISION rate > Λ0;
     (T):
      TASK ci := T, rate := Λ0;
     (F):
     ENDDECISION;
    (F):
     COMMENT 'other messages are ignored';
    ENDDECISION;
   (Ack):
    DECISION tf;
    (T):
     COMMENT 'Acknowledgement for a traffic message
              initiated by the ingress. The node shifts to
              Waiting state so that error recovery can
              be performed without hack';
     OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
     NEXTSTATE Waiting;
    (F):
    ENDDECISION;
   ENDDECISION;
   OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
   NEXTSTATE -;

ENDPROCESS PInt;
```

A.3.3 Egress node

An egress node has three states as an intermediate node. The procedures used by an egress node are similar to the procedures used in an ingress node. The pseudo code on the egress node is given in table A.5.

Table A.5: Pseudo code of the egress node

```
PROCESS Egress;

  SYNONYM lmax Bandwidth = EXTERNAL;
  SYNONYM lpcr Bandwidth = EXTERNAL;
  SYNONYM $T_{Egress}$ Duration = EXTERNAL;

  DCL Λ,Λ’,rate Bandwidth;
  DCL ign_seq,seq,seqcpt Integer;
  DCL kind Kind;
  DCL ci,el,mt,tf ,el_memory,aux Boolean;
  TIMER $W_{Egress}$;

  PROCEDURE Book REFERENCED;
  PROCEDURE change_number REFERENCED;
  PROCEDURE attribute_msg REFERENCED;
  PROCEDURE check_number REFERENCED;
  PROCEDURE SETNumbering REFERENCED;

  START ;
    TASK Λ := Null, Λ’ = Null;
    NEXTSTATE Normal;

  STATE Normal;
    INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
      DECISION kind;
      (Req):
        DECISION tf;
        (T):
          COMMENT 'Traffic request from the ingress';
          EgressNetTrafficFwd;
          CALL Book(Λ,rate);
          COMMENT 'On a UNI loop it back';
          CALL SETNumbering;
          TASK el_memory := el;
          SET(NOW+$T_{Egress}$,$W_{Egress}$);
          OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq)  VIA UpS;
          TASK kind := Ack;
          OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq)  VIA DoS;
          NEXTSTATE Traffic;
        (F):
          DECISION el or (not(ci) and rate > Λ);
          (T):
            COMMENT 'Successful user BCR increase';
            CALL Book(Λ,rate);
            OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq)  VIA UpS;
            COMMENT 'Loop back on the UNI';
            TASK kind := Ack, el_memory := el;
            OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq)  VIA DoS;
            SET(NOW+$T_{Egress}$,$W_{Egress}$);
            NEXTSTATE Waiting;
          (F):
            DECISION rate <= Λ;
            (T):
              COMMENT 'User BCR decrease';
              CALL Book(rate,Null);
              OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq)  VIA UpS;
              NEXTSTATE Normal;
            (F):
              COMMENT 'Unsuccessful BCR modification';
              CALL Book(Λ,Null);
              COMMENT 'Loop back on the UNI';
              CALL Book(Λ,Null);
              TASK kind := Ack;
              OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq)  VIA DoS;
              NEXTSTATE Normal;
            ENDDECISION; /* rate ≤ Λ */
          ENDDECISION; /* el or not(ci and rate \(>\) Λ) */
        ENDDECISION; /* tf */
      (Ack):
        COMMENT 'No ack in this state’;
        NEXTSTATE Normal;
      ENDDECISION; /* kind */

    INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
      EgressBackward:
      TASK rate := min(rate,lmax);
      DECISION kind;
      (Req):
        EgressBackwardReq:
        DECISION tf;
        (T):
          COMMENT 'Reset the tf bit on a user request';
          TASK tf := F;
          JOIN EgressBackwardReq;
        (F):
          COMMENT 'User request initialization';
          TASK ci:=F;
          DECISION mt;
          (T):
            COMMENT 'Initialize status enquiry';
            TASK rate := Λ, el := T;
          (F):
            COMMENT 'Initialize BCR modification';
            TASK el:= IF rate <= Λ THEN F ELSE el FI;
          ENDDECISION;
          CALL Book(Λ,rate);
          SET(NOW+$T_{Egress}$,$W_{Egress}$);
          TASK el_memory := el;
          OUTPUT BMsg(kind,rate,ci,el,mt,tf,seq) VIA DoS;
          NEXTSTATE Waiting;
        ENDDECISION; /* tf */
      (Ack):
      ENDDECISION; /* kind */
      NEXTSTATE -;

    INPUT POL(el);
      COMMENT 'Policing action from the egress';
      EgressPolicing:
      TASK rate := IF el THEN lpcr ELSE Λ FI;
      TASK el_memory := el;
      SET(NOW+$T_{Egress}$,$W_{Egress}$);
      CALL Book(Λ,rate);
      CALL change_number;
      CALL attribute_msg;
      OUTPUT BMsg(kind,rate,ci,el,F,tf,seq) VIA DoS;
      NEXTSTATE Traffic;

  STATE Waiting;

    INPUT $W_{Egress}$;
      COMMENT 'The TIMER has expired';
      TIMEREgress:
      CALL change_number;
      CALL attribute_msg;
      SET(NOW+$T_{Egress}$,$W_{Egress}$);
      OUTPUT BMsg(Req,Λ’,F,el_memory,T,T,seq) VIA DoS;
      NEXTSTATE Traffic;

    INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
      DECISION kind;
      (Req):
        COMMENT 'Only traffic requests are treated';
        DECISION tf;
        (T):
          RESET($W_{Egress}$);
          JOIN EgressNetTrafficForward;
        (F):
          NEXTSTATE -;
        ENDDECISION; /* tf */
      (Ack):
        COMMENT 'Acknowledgement (New block) from the source';
        CALL Book(rate,Null);
        RESET(W_Egress);
        OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
        NEXTSTATE Normal;
      ENDDECISION; /* kind */

    INPUT BMsg(kind,rate,ci,el,mt,tf,seq);
      COMMENT 'Only traffic requests from an upward network
              are treated otherwise sends a Not Ready';
      OUTPUT FMsg(Ack,rate,T,T,T,F,seq) VIA UpS;
      NEXTSTATE -;

    INPUT POL(el);
      JOIN EgressPolicing;

  STATE Traffic;

    INPUT W_Egress;
      JOIN TIMEREgress;

    INPUT FMsg(kind,rate,ci,el,mt,tf,seq);
      DECISION kind;
      (Req):
        JOIN EgressForwardRequestInTraffic;
      (Ack):
        COMMENT 'Treat Acks from the source with the right
                sequence number';
        DECISION tf;
        (T):
          CALL check_number(aux);
          DECISION aux;
          (T):
            RESET(W_Egress);
            CALL Book(rate,Null);
            OUTPUT FMsg(kind,rate,ci,el,mt,tf,seq) VIA UpS;
            NEXTSTATE Normal;
          (F):
          ENDDECISION; /* aux */
        (F):
        ENDDECISION; /* tf */
        NEXTSTATE -;
      ENDDECISION; /* kind */

    INPUT BMsg(kind,rate,ci,el,mt,tf,ign_seq);
      OUTPUT FMsg(Ack,rate,T,T,T,F,seq) VIA UpS;
      NEXTSTATE -;

ENDPROCESS Egress;
```

A.4 Further considerations

In the above specification, a protocol for supporting the ABT/DT capability has been presented. Note that the selection of the parameters of the protocol (W_Ingress, W_Egress, T_{RM}, T'_{RM}, etc.) has not been discussed. In fact, the implementation of the timers at network ingress and egress is not necessary if BCR negotiations are performed periodically by the network, in which case either user does not need to initiate any BCR negotiation and has just to acknowledge the Bandwidth Acknowledgement RM cells sent by the network. This is a possible use of the above protocol for which some options are removed.

History

Document history		
V1.1.1	November 1997	Publication

ISBN 2-7437-1825-0
Dépôt légal : Novembre 1997