The Compound VC switch. A non-VC merge ATM multicast switch.[*]

Josep Mangues-Bafalluy and Jordi Domingo-Pascual
Technical University of Catalunya (UPC)
Advanced Broadband Communications Center (CCABA)
Computer Architecture Department
Campus Nord. Mòdul D6. Sala 008.
Jordi Girona 1-3. 08034 Barcelona (Spain)
{jmangues | jordi.domingo}@ac.upc.es

Abstract – VC merge has been presented as the most likely implementation to offer multicast forwarding in MPLS environments when ATM is used as data link layer. But it presents some characteristics that may not be desirable, like extra buffering and delay, and modification of traffic characteristics. Other types of mechanisms are presented in the literature to solve these problems. Compound VC (CVC) is one of these mechanisms. This paper presents a general comparison of VC merge and CVC. It finally discusses the implementation issues of a CVC switch. Though its implementation is more complex, it is shown that CVC forwarding could be carried out by using state-of-the-art ATM hardware.

I. INTRODUCTION

AAL5 is the most commonly used adaptation layer for ATM networks. Just one bit in the cell header of the last cell allows the switch to delineate AAL5 PDUs. There is no way of knowing which PDU a given cell belongs to. Thus, multiplexing of cells belonging to different AAL5 PDUs must be avoided, or solved in some way, in order to reassemble them correctly at the end-system. This problem is known as the cell-interleaving problem.

IP multicasting over ATM [1] proposes some solutions to solve the problem, but they present inefficiency in terms of signaling overhead or delay depending on the chosen strategy. Native ATM multicasting mechanisms try to solve this inefficiency by solving the cell-interleaving problem at the ATM layer. That is, their goal is to offer layer 2 multicast forwarding. In this paper, multicast is understood as the provisioning of multipoint-to-multipoint communications.

The interest of such mechanisms comes from the fact that many vendors have used ATM to implement MPLS-capable routers. Thus, these techniques could help in provisioning the multicast capability for ATM Label Switch Routers (ATM-LSRs).

Some native ATM mechanisms have been proposed in the literature to solve the cell-interleaving problem at the ATM layer. They may be classified into four groups. The first group solves the problem by avoiding cell interleaving to happen by means of either buffers or token passing protocols. The second group uses the VPI to identify the group and the VCI to identify either the sender or the PDU. The third group provides mechanisms to allow the multiplexing of cells inside a single VC either by adding the multiplexing overhead in the transmitted data, or by using the GFC field for that purpose, or by negotiating, at connection establishment, the order in which cells are going to be transmitted. Finally, the fourth group allows interleaving of cells by using 2 or more VCIs for the same multicast group. A more detailed review of these mechanisms may be found in [2].

The mechanism that has received more attention in the literature is VC merge [3] (also referred to as store-and-forward in [2]) due to its implementation simplicity. It belongs to the first type of the above classification, and avoids cell-interleaving by using additional buffers. However, this simplicity comes at the price of additional delay, delay variation, and buffering. Thus, traffic characteristics are changed.

Another mechanism called Compound VC (CVC) [4] was introduced to solve the problems that appear with VC merge at the price of using more VCI space. A comparison of both mechanisms based on previous work is presented.

The goal of this paper is to present the functional architecture of a CVC switch. The simple implementation of CVC may help in the provisioning of multicast communications with quality of service in MPLS environments using ATM-LSRs.

The next section is devoted to compare VC merge and CVC. The following sections present the operation and characteristics of a generic ATM switch as well as a VC merge switch. Section V introduces the CVC forwarding operation by means of an example, and also presents the functional architecture of a CVC switch. Finally, we present the conclusions.

II. VC MERGE VS. COMPOUND VC

A. *VC Merge*

VC merge (or store-and-forward) [3] avoids cell-interleaving by storing all the cells of a PDU in a buffer and by forwarding all these cells in an atomic manner once the last cell of the PDU arrives. In this sense, a VC merge switch emulates frame switching of AAL5-PDUs without carrying out the AAL reassembly process.

B. *Compound VC*

The main idea in the Compound VC (CVC) mechanism [4] is that the multicast group is associated to a group of adjacent VCs (or compound VC). The VCI field in the ATM cell header is divided into two parts. The first one corresponds to the CVC ID and the other one carries the PDU ID (Fig. 1). Reference [5] shows the advantages of using PDU IDs instead of Source IDs.

Therefore, the tuple (VPI, CVC ID) identifies a group in a unique way and the least significant bits of the VCI field identify the AAL5 PDU.

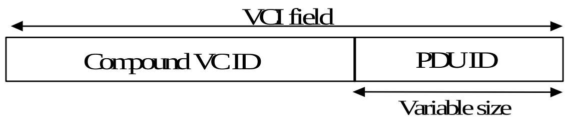

Fig. 1. VCI field in Compound VC

[*] This work has been funded by the Spanish Ministry of Education (CICYT) under grant number TEL99-1117-C03-03 (SABA-2)

The VPI, CVC ID and PDU ID are locally mapped at the switches. The CVCID mapping entries do not change during the connection, while the PDU ID mapping entries are dynamically modified each time the first cell of a new PDU traverses the switch or each time the last cell is transmitted.

Besides, the size of each part is negotiated at connection establishment. This feature allows more flexibility in matching different group sizes and traffic characteristics.

C. *Comparison of VC Merge and CVC*

Simulation results [4] show that throughput obtained by both mechanisms is the same if the same number of reassembly buffers for VC merge as PDUIDs for CVC is chosen.

The main difference is the behavior of the pattern of the outgoing traffic. VC merge includes buffering and transmits the cells of the same PDU grouped and at the peak rate. On the other hand, CVC simply multiplexes the arriving cells without increasing neither the end-to-end delay nor the cell delay variation. The price paid is extra VCI consumption. But the number of additional VCIs is not high. As shown in [5], a few PDU IDs are enough to serve groups with high number of sources for the simulated traffic conditions.

The graphical example of the behavior of both mechanisms presented in [2] could help in showing their influence on traffic characteristics. This point is further analyzed in [6], where comparisons concerning the delay behavior of some mechanisms are carried out. In particular, there is a comparison between VC merge (referred to as hardware VC merge in that paper) and non-hardware VC merge. As far as delay in the buffers is concerned, the results obtained for non-hardware VC merge also apply to Compound VC (CVC). The only difference among both may appear due to processing times at the nodes. However, they are likely to be small compared to the buffering delay because the processing is carried out with hardware, as we explain below. The interesting part of that work, was the utilization of real traffic traces. The results showed that the delay for hardware VC merge was 65% higher across the range of network load until 85%. Standard deviation also was 95% higher in the hardware VC merge case, and this increase stayed approximately constant up to about 85% utilization.

This aspect may not be very important for data transmission but may significantly distort the traffic characteristic of real-time multimedia communications, which are common among multicast applications.

Other results obtained through simulation and analytically also show that VC merge requires more buffers than non-VC merge mechanisms [7], though in some cases, and for the studied traffic, the authors claim that this buffer increase is not significant. However, there are other studies claiming that the conclusions of that paper are in contrast with their findings [8].

Anyway, additional buffering is not the only concern when dealing with VC merge. Quality of service (QoS) requirements must also be satisfied. In the same paper [7], the authors propose some modifications in VC Merge to offer per-class quality of service. The paper suggests using different output buffers for traffic requiring different QoS. Each output buffer is associated with a different VC and a cell level scheduling mechanism is responsible for interleaving cells of different classes in order to minimize traffic distortion. As a consequence, buffering requirements and VCI consumption increase.

With respect to delay, and apart from that introduced by the extra buffering, one could expect that delay and cell delay variation somehow increase when multiple merging nodes are crossed when VC merge is used.

Signaling requires some modifications if CVC is to be supported because the establishment of the connection must handle a group of VCIs as a compound VC. Interoperability issues with standard signaling must also be considered. Some ways to deal with interoperability may be found in [5].

Finally, with respect to implementation, VC merge is simple. However, CVC may be implemented with slight modifications to state-of-the-art ATM hardware. The implementations of both mechanisms are compared in the following sections.

III. Functional blocks of an ATM switch

The generic architecture of an ATM switch that we take as a reference for the discussion below is represented in Fig. 2.

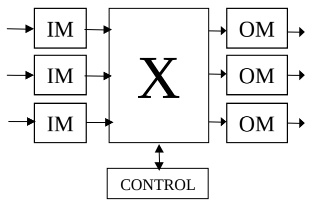

Fig. 2. Generic switch block diagram

The control block is in charge of dealing with the signaling and management operations. The input module (IM) receives incoming cells, maps their VPI/VCI, and forwards them to their corresponding output module/s. For that purpose, it appends additional information to each cell in the form of tags. These internal tags are used by the switch fabric to route the cells to the output modules (OMs). Finally, the OM buffers the cells coming from the IMs and sends them to the output link after having formatted them appropriately. The following sections describe how forwarding is carried out in a VC merge switch and a CVC switch.

IV. Multicast Forwarding with VC merge

A. *Traffic forwarding*

The forwarding in a multicast group using VC Merge is carried out through the only established VC, i.e. in principle, there is just one VC per group. The cell-interleaving problem is solved by the output module of the switch. It is in charge of buffering all the cells of a given PDU and sending them together once they were all received. Fig. 3 shows an example of VC merge forwarding.

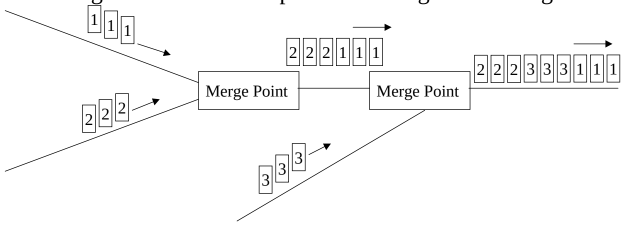

Fig. 3. VC merge forwarding example

B. *Block functionality description*

The main difference in a VC Merge switch with respect to conventional switches is in the output module. All the required buffering management is carried out there. Fig. 4 presents a block diagram of such an output module. Only ATM cell forwarding parts are represented. SONET/SDH operations, signaling and management cell insertion, etc. are not represented.

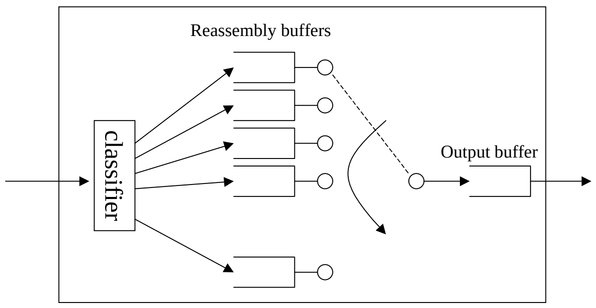

Fig. 4. Output module of a VC merge switch

When cells arrive to the OM after having crossed the switch fabric, they are classified and put into their reassembly buffer according to the tuple (input module, VCI). Remark that the IM must add an internal tag to allow the OM to differentiate between cells coming from different IMs with the same VCI [7]. The IM number is carried in an internal tag. Once all the cells of a PDU arrive, they are moved in an atomic manner to the output port from where they are transmitted through the output link.

V. MULTICAST FORWARDING WITH CVC

Fig. 5. shows a scenario where CVC could be used. It is homogeneous in the sense that all users in the multicast group (or CVC connection) receive the same treatment and introduce the same traffic to the network. A typical example of such a scenario is a fully interactive videoconference.

A. Traffic forwarding

Once the multicast tree has been established with the right number of IDs assigned to the CVC connection, the forwarding of traffic begins. We assume all switches in the network perform CVC forwarding.

When a sender wants to transmit to the group, it sends its information through the single VC that connects it to the ingress switch. Once in the switch, if there is no traffic from the group being forwarded to the same output link, the packets will be forwarded without PDU ID remapping, i.e. just the compound VC ID mapping part is required. For instance, when sender A in Fig. 5. wants to transmit, it sends its information through link A-S1 with the only ID assigned to this link for this multicast group. In switch S1, there is no information from other members of the group being forwarded from S1 to S2. Thus, PDU ID mapping in S1 is not required and the packet is forwarded by just looking at the CVC ID switching table.

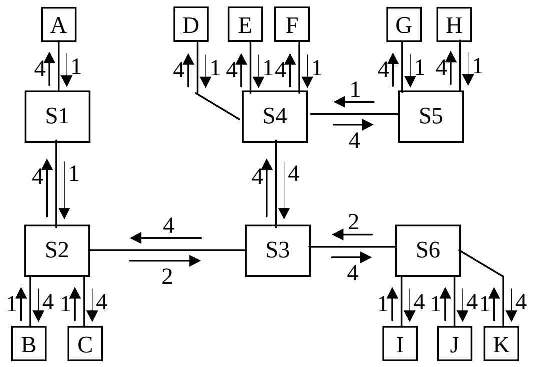

Fig. 5. Example of multicast scenario with CVC

If there are other senders forwarding its information through the same switch, both flows will be aggregated. Therefore, the CVC forwarding mechanism will be used to map the PDU IDs. For instance, when the PDUs sent by sender A arrive to switch S2, they are multiplexed with those coming from senders B and C and they are forwarded from S2 to S3 through a 2-ID compound VC.

Fig. 5. also shows that the number of IDs assigned to the group in a given link may not be the same in both directions. It depends on the characteristics of the aggregated traffic and will be calculated at connection establishment. For example, the link connecting a switch with a member of the group uses 1 ID in the ingress direction, because just this member uses it. But it uses 4 IDs in the egress direction because the aggregated traffic of the whole group is forwarded to all the components of the multicast group.

B. Block functionality description

The previous subsection showed the functionality offered by CVC for multicast forwarding at the group level. This subsection deals with the ATM-level block architecture inside the switches to offer the CVC forwarding capability.

There are two options for the implementation of the CVC switch, namely the Longest Prefix Match (LPM) and the 2-Mappings (2MAP). The first one allows to treat the group as a whole by just having one entry for any given group in the CVCID mapping table of the IM. This implementation involves LPM lookup operations because the CVC ID is variable in length. LPM lookups are costly to implement in hardware. On the other hand, in the 2MAP implementation, the switch must carry out two mappings of the VCI field of each cell. But these mappings are carried out with indexing table lookup operations. This paper focuses on 2MAP because it is simple and it may be implemented with state-of-the-art ATM hardware.

As presented in section II, in CVC, the VCI field is divided into two parts, a CVC ID part identifying the multicast group and a PDU ID part identifying the packet to which a cell belongs. Both identifiers are locally remapped at each switch. The following sections explain how each of these mappings is carried out by means of a CVC ID mapping table and the Dynamic VCI Assignment (DVA) respectively. The example presented in the appendix (Fig. 7 and tables) will serve us to explain this implementation.

1) Input Module – CVC ID mapping

The IM is in charge of the mapping of the CVC ID. This mapping, for a given CVC connection, is determined at connection establishment. That is, the entries in the CVC ID mapping table for a given group are filled in when the connection is established and are removed from the table when the multicast connection ends. The number of entries in the mapping table of a given IM corresponding to a single CVC connection is equal to the number of IDs used in the input link to that IM. In our example Fig. 7, 2 IDs (and thus, 2 entries in the table) for IM 1, 4 IDs (4 entries) for IM 2, and 4 IDs for IM 3.

These entries are used to map the CVC ID field of the incoming cells and to forward them to the appropriate output port/s. The 2MAP implementation treats the entries in the IM table as if they were normal VCIs. Thus, state-of-the-art ATM hardware could be used. This is in contrast with the LPM implementation, which just requires one entry in the table for each group. But the price paid is major changes in ATM hardware.

Notice that the PDU ID is also mapped jointly with the CVCID when the whole VCI is changed. But the PDU ID will only be paid attention in the OM.

The range of VCIs allocated to the output link will determine the CVCID and PDU ID values to which incoming cells are mapped. In our example, the output VCIs allocated for the group at OM 4 are comprised in the range 0xAF28 to 0xAF2F (Table 4). This range is chosen at connection establishment and the OM sends to each IM involved in the forwarding of the CVC connection a number of VCIs inside this range equal to the number of input IDs to that IM. These VCIs will be used to fill in the output VCI column of the CVC ID mapping table. For instance, OM 4 sends IDs 0xAF28 and 0xAF29 to IM1, which is assigned 2 IDs

Once the CVC ID mapping is done, a routing tag is added to the cell. This tag will allow the switch fabric to route the cell to the correct output port. If there is a single destination port, the tag will directly contain the output port number. But if there are multiple destination ports, this tag will carry a multicast address that will allow the multicast-capable switch fabric to duplicate and route the cell to the corresponding ports [9].

A second tag is also added to the cell. It contains the IM through which the cell arrived to the switch. This field is required in case that cells coming from different IMs with the same VCI, and thus the same PDU ID, must be forwarded to the same OM.

Therefore, in most operations, the IM behaves exactly like a normal ATM switch. The only slight difference is the inclusion of a second information tag next to the routing tag, which is also required for VC merge hardware [7]. This means that state-of-the-art hardware could be used in the Input Module with slight modifications.

2) *Output Module - Dynamic VCI Assignment (DVA)*

Fig. 6 presents the internal structure of the cell-processing block of the Output Module of a CVC switch. It is based on the functional blocks described in [9]. Only blocks used for handling CVC cells at the ATM level are represented. Other blocks for signaling and management cells handling are not represented.

The OM is assigned the task of PDU ID mapping to avoid ID collision when traffic coming from different input ports is multiplexed. Therefore, if such multiplexing does not occur, the DVA block functionality is not required, and cells are forwarded with just CVC ID mapping.

When a cell arrives to the OM after being routed by the switch fabric, it arrives to the cell-processing block. PDU ID mapping is carried out in the Dynamic VCI Assignment (DVA) block (Fig. 6). Notice that though the PDU ID is variable in size, its mapping is carried out by changing the whole VCI (Table 4). Thus, the OM just handles fixed-length fields. Notice also that though we map the entire VCI, the only thing that is changed in the OM is the PDU ID because the CVC ID was already changed in the IM.

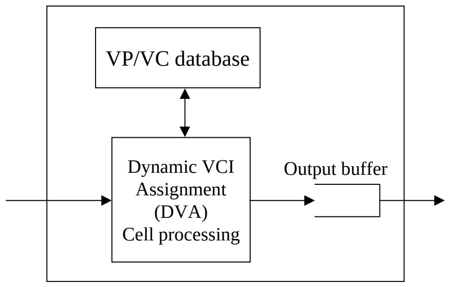

Fig. 6. Cell processing block of output Module of a CVC switch

Therefore, when the cell enters the cell-processing block, the DVA looks up the VP/VC database and translates the VCI field in the cell header depending on the tuple (input module, input VCI) (Table 4). The output VCI carries a unique PDU ID for that CVC connection. That is, no other PDU belonging to the same CVC connection and passing through the same output module is assigned that ID.

Once the translation is done, the internal tags are removed and the cell is stored in the output buffer, from where it will transmitted to the link.

The actions to be undertaken when mapping a cell depend on the type of cell inside an AAL5-PDU (initial, middle or last). When the initial cell arrives, a free PDU ID is assigned to the PDU and the PDU ID is marked busy to avoid other PDUs from using it. If all PDU IDs are already being used by other PDUs, early packet discard (EPD) is applied to the current PDU (e.g. first entry of Table 4).

If middle cells must not be discarded, once they enter the DVA block, the (input module, input VCI) tuple is searched in the table and the output VCI (which includes the PDU ID) of the matching entry is written in the cell header (Table 4).

The binding between the input tuple and the output PDU ID lasts until the last AAL5 cell arrives, i.e. once the last cell arrives, the same mapping process is carried out, but after that, the output PDU ID is freed and the entry is removed from the table.

Once again, the implementation is carried out by mapping the whole VCI instead of just the PDU ID with the aim of preserving most operations of a normal ATM switch. In this way, with only slight modifications in VCI table processing, it is possible to offer multipoint-to-multipoint connections without degradation of traffic characteristics. Besides, the OM of some current multicast (point-to-multipoint) switches may require cell processing blocks to map multicast connection IDs to output VPI/VCI values [9]. Therefore, in this case, there won't be any extra hardware apart from the one that controls the actualization of the VP/VC database each time a PDU starts or ends.

Thus, and as a concluding remark, notice that in the 2MAP implementation, the switch is capable of mapping two variable length fields (CVCID and PDUID) by performing two fixed-length-mapping operations. As a consequence, the CVC mechanism may be implemented with minor modifications to current ATM hardware and without loosing the flexibility of the CVC mechanism.

VI. CONCLUSIONS

We have presented a general comparison of two mechanisms (VC merge and Compound VC) to offer multicast capability in ATM environments while solving the cell-interleaving problem. The wide acceptance of ATM as data link layer for MPLS justifies the studies around multicasting at the ATM layer.

Previous work shows that VC merge requires more buffering than mechanisms allowing cell interleaving. Furthermore, apart from the delay introduced by this buffering, there are additional effects on the traffic characteristics, which could make quality of service difficult to achieve.

On the other hand, CVC tries to solve all these problems at the price of a slightly more complex implementation. This paper presented an implementation of a CVC switch that allows the utilization of state-of-the-art ATM hardware while preserving the traffic characteristics.

REFERENCES

[1] Armitage, GJ 'IP multicasting over ATM Networks.' IEEE Journal on Selected Areas in Communications 15(3): 445-457, April 1997.

[2] Mangues-Bafalluy J and Domingo-Pascual J. 'Multicast forwarding over ATM: Native approaches.' IEEE Communications Surveys, 3rd Quarter 2000.

[3] Rosen EC, Viswanathan A, and Callon R. 'Multiprotocol label switching architecture.' IETF RFC 3031. January 2001.

[4] Mangues-Bafalluy, J. and Domingo-Pascual, J. 'Compound VC Mechanism for Native Multicast in ATM Networks.' Proceedings of the 2nd International Conference on ATM (ICATM'99). Colmar (France), June 1999.

[5] Mangues-Bafalluy J and Domingo-Pascual J. 'Performance Issues of ATM Multicasting based on Per-PDU ID Assignment.' Proceedings of IEEE International Conference on Communications (ICC'00). New Orleans (USA), June 2000.

[6] Boustead P, Chicharo J, and Anido G. 'Scalability and performance of label switching networks.' Proceedings of GLOBECOM'98, pp. 3029-3034, 1998.

[7] Widjaja I and Elwalid AI. 'Performance Issues in VC-Merge Capable Switches for Multiprotocol Label Switching.' IEEE Journal on Selected Areas in Communications 17(6): pp. 1178-1189, June 1999.

[8] Zhou P and Yang OWW. 'Reducing buffer requirement for VC-merge capable ATM switches.' Proceedings of GLOBECOM'99, pp. 44-48, 1999.

[9] Chen TM and Liu SS. 'ATM switching systems.' Artech House Publishers, 1995.

APPENDIX: EXAMPLE OF FORWARDING IN A CVC SWITCH

An example of the operation of CVC when multiplexing in a switch occurs is described in the following paragraphs. First, Fig. 7 describes the scenario. The mapping tables related to it are also shown below. The first three tables describe the CVCID mapping tables at the IM1, IM2, and IM3 for a given CVC connection. For example, the traffic belonging to this group enters input module 2 with VPI=3, CVCID=01110011010000 (these bits correspond to the more significant bits of the VCI field). As we have two bits assigned to the PDU ID, there are four possible identifiers. That is why there are four inputs in

Table 2. The same considerations apply for the rest of the tables of the input modules.

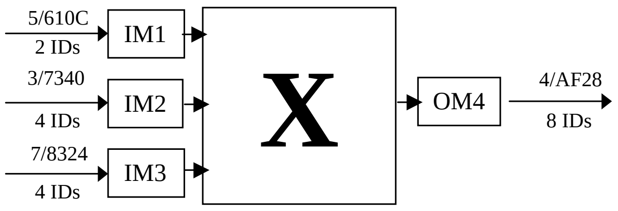

Fig. 7. Cell forwarding in a CVC switch.

For all tables, just entries corresponding to the CVC connection in Fig. 7 are shown.

Table 1: CVC ID mapping table at input module 1

Input		Output		
VPI	VCI	VPI	VCI	OM
5	610C	4	AF28	4
5	610D	4	AF29	4

Table 2: CVC ID mapping table at input module 2

Input		Output		
VPI	VCI	VPI	VCI	OM
3	7340	4	AF28	4
3	7341	4	AF29	4
3	7342	4	AF2A	4
3	7343	4	AF2B	4

Table 3: CVC ID mapping table at input module 3

Input		Output		
VPI	VCI	VPI	VCI	OM
7	8324	4	AF28	4
7	8325	4	AF29	4
7	8326	4	AF2A	4
7	8327	4	AF2B	4

Table 4 corresponds to the PDU ID switching table at output module 4. In this example, all the cells coming from IM1, IM2, and IM3 have been given a CVCID equal to 1010111100101, and the three less significant bits in the VCI field of the OM are used as PDU ID. Thus, there are 8 IDs. But, there may be some cases in which all the identifiers of all the input modules are being used. In this example, ten simultaneous PDUs are trying to pass through OM4 (2IDs from IM1 + 4IDs from IM2 + 4IDs from IM3). As represented in Table 4, all the cells of two PDUs are discarded due to having run out of output identifiers.

Table 4: PDUID mapping table at output module 4

input		output	
VCI	IM	VCI	discard
AF28	1	-	1
AF29	1	AF28	0
AF28	2	AF29	0
AF29	2	AF2A	0
AF2A	2	AF2B	0
AF2B	2	AF2C	0
AF28	3	-	1
AF29	3	AF2D	0
AF2A	3	AF2E	0
AF2B	3	AF2F	0