MOTOROLA intelligence everywhere

C-Ware DSLAM Uplink Networking Line Card Application

User Guide

CALDSLAM-UG/D
Version 1.0, 11/25/2002

1. Introduction

1.1 Purpose of Document

This document presents the software architecture and design for a DSLAM Uplink Networking Line Card reference application that is targeted for Motorola's C-5 and C-5e network processors.

1. 2 Software Architecture Overview

The C-5/C-5e network processors can be divided into four clusters of four channel processors each, where each cluster is typically used to provide high speed and/or high processing neworkingfunctions across all four processors. In case of a DSLAM Uplink Networking Line Card, only 3 out of 4 clusters are required for the application as shown in Figure 1.

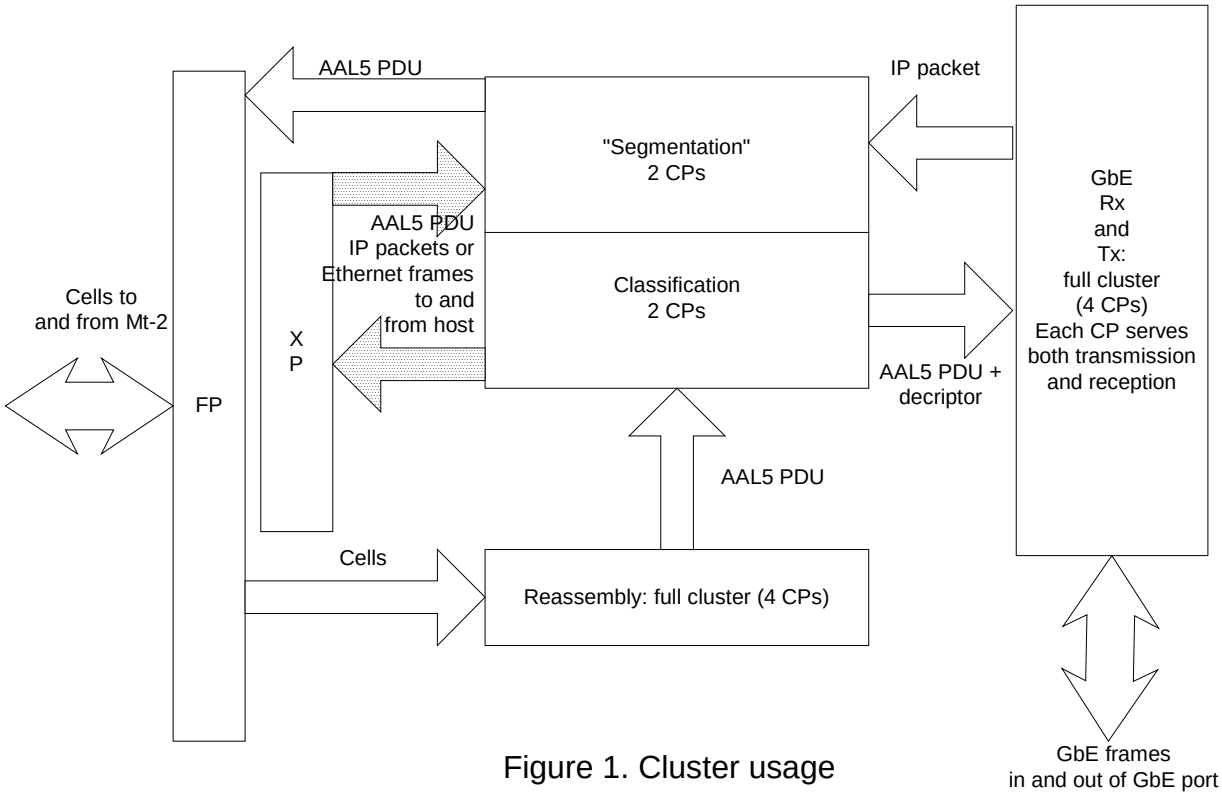


Figure 1. Cluster usage

DSLAM Uplink Networking Line Card application has 2 ports: FP, which can be connected to the cell bus or switch fabric, and Gigabit Ethernet. The application performs the following functionality:

- IPv4 forwarding
- L2TP tunneling
- PPP over Ethernet (FP side)
- VC-multiplexing and LLC-encapsulation for AAL5 frames (FP side)
- Forwarding the AAL5 PDUs and IP packets in and out of the host (XP-based host interface is not implemented in this scenario)

There are 3 clusters that are used. One of the clusters performs two completely separate kinds of functionality. 2 CPs perform classification, that is, processing of AAL5 PDUs received from FP. The other 2 CPs convert IP packets into AAL5 PDUs while they travel from GbE port to FP
There is also an executive processor (XP) and a fabric processor (FP) present on the network processor.

1.3 Data flow overview

The DSLAM application supports a number of different types of traffic. Traffic flowing from FP to Gigabit is called **upstream**. Traffic flowing from Gigabit port to FP is called **downstream**. Traffic flowing to the host is called **host-bound**. Traffic flowing from the host is called **host-generated.**
All cells coming in and out of FP belong to a particular Virtual Circuit. Every cell has 4-byte header, which contains 24-bit VPI/VCI that determines VC. 4-byte header is not a part of a cell body. When the cell comes into the FP, cell header is stripped and saved in a descriptor. When the cell goes out of FP, descriptor is added to the cell.
Only AAL5 is supported by the DSLAM Uplink Networking Line Card application in this example.

1.3.1 Upstream traffic

The following types of traffic (AAL5 frames) are supported through the DSLAM Uplink Networking Line Card application, processed and passed to the GbE port:

- IP over AAL5, both VC-multiplexing and LLC-encapsulation. Forwarded from GbE port as an IP packet
- PPP over Ethernet over AAL5, both VC-multiplexing and LLC-encapsulation. Goes out of GbE port as L2TP frame, encapsulated within UDP datagram
- PPP over AAL5, both VC-multiplexing and LLC-encapsulation. Goes out of GbE port as L2TP frame, encapsulated within UDP datagram

Each VC has a type of traffic assigned to it. For a particular VC, DSLAM application passes only those frames that match an exact traffic definition of that VC. Otherwise, the frame is dropped.
Upstream traffic is shown on a picture 2.

1.3.2 Downstream traffic

The following types of processing are supported on GbE port:

- IP forwarding
- L2TP frame tunneling

DSLAM application selects the processing mode based on IP DA, source and destination UDP port
Downstream traffic is shown on a picture 3.

1.3.3 Host-bound traffic

Some frames received on a FP are forwarded to the host. Among them are:

- Host-bound IP over AAL5 frames (IP DA = internal IP addr)
- PPP over Ethernet discovery frames
- PPP control frames, in case when the session hasn’t been established yet
- IP over AAL5 frames, in case when the route has been established but not resolved through ARP

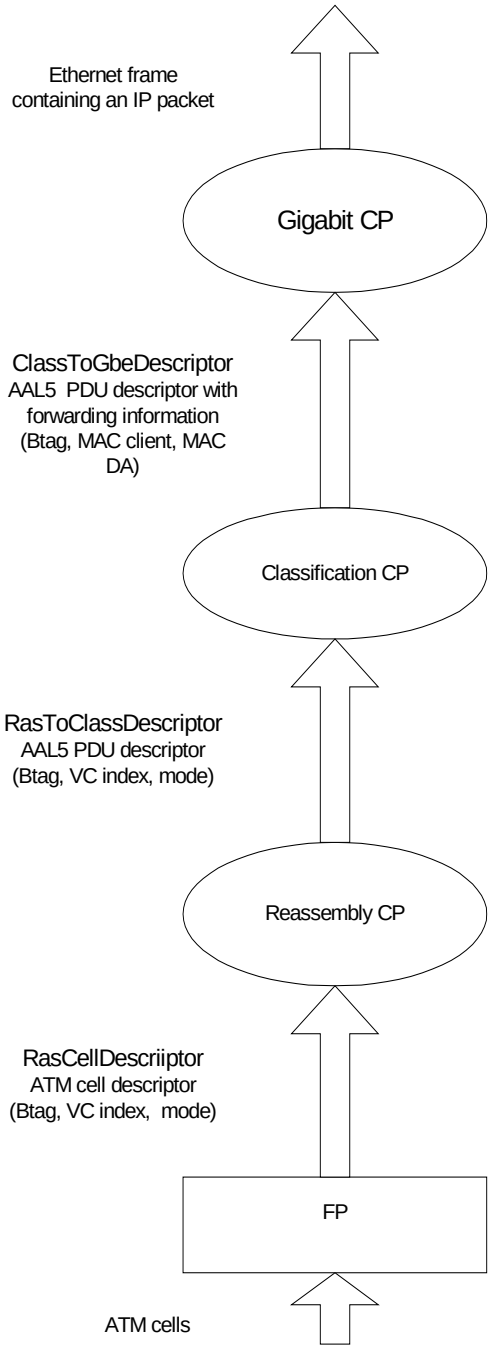
Host-bound traffic is shown on a picture 4.

Some Ethernet frames and IP packets received on GbE port are also forwarded to the host. Among them are:

- Ethernet ARP frames
- Host-bound IP packets
- L2TP control packets

1.3.4 Host-generated traffic

DSLAM Uplink Networking Line Card application is not concern with the kind of frame the host wants to send to FP. It just has to be a valid AAL5 frame. Likewise, DSLAM Uplink Networking Line Card application will forward to GbE port any Ethernet frame or IP packet generated by the host
The host enqueues IP packet and Ethernet frame descrptors directly into the input queue of GbE cluster
Likewise, the host enqueues the AAL5 PDU descriptor directly into the input queue of FP.



Gigabit CP (TxByte):

Strips LLC header if it present	**All**
Adds Ethernet MAC header	**All**
Adds IP header, UDP header and L2TP header	**L2TP**
Strips AAL5 trailer and calculates Ethernet CRC	**All**

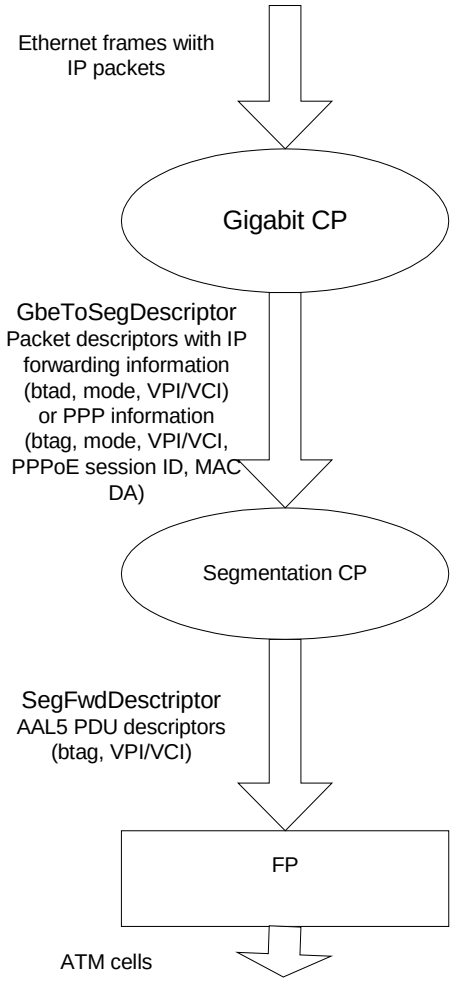
Classification CP sends an incoming AAL5 PDU through the SDP recirculation path

RxByte verifies LLC header and AAL5 PDU CRC	**All**
launches IP DA lookup	**IP**
launches PPPoE session lookup	**PPPoE**
launches VC index lookup	**PPP**
calculates the payload checksum	**PPP and PPPoE**
RC obtains:	
the IP forwarding information (MAC DA)	**IP**
the L2TP forwarding information (MAC DA, IP DA, UDP dest port, L2TP tunnel/session	**PPP and PPPoE**

Reassembly CP builds an entire AAL5 PDU out of incoming cells. It keeps the reassembly context in DMEM for each AAL5 PDU currently being reassembled. To attach a new cell to the PDU, CP sends a cell through the SDP recirculation path

FP strips 4-byte header from the incoming cel
Launches VPI/VCI lookup and obtains VC index and processing mode for that VC

Picture 2. Upstream traffic



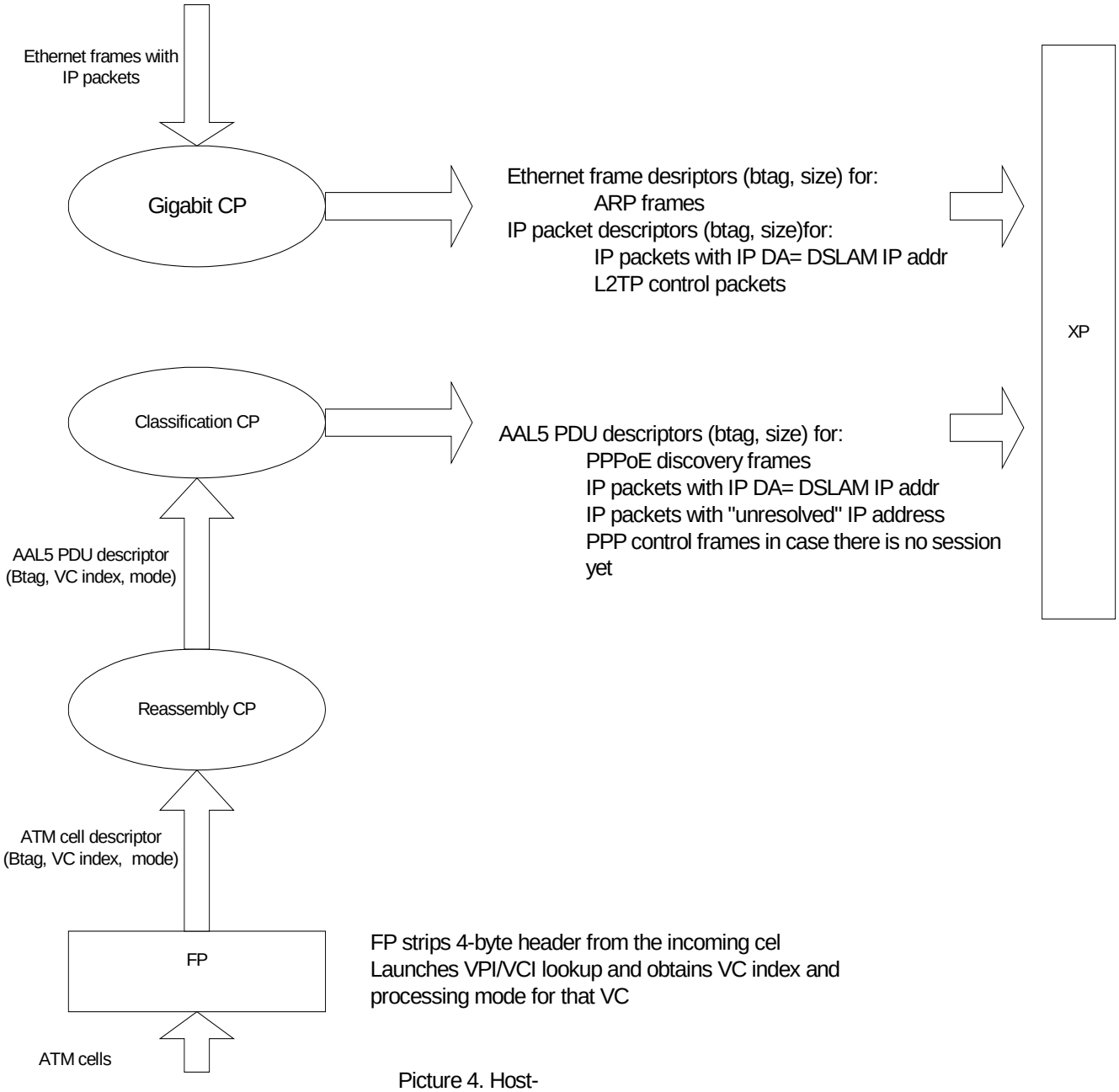
Gigabit CP receives an ethernet frame with IP packet
RxByte CP launches IP DA lookup
If IP DA != internal IP DA, try IP routing
If IP DA == internal IPDA, try L2TP tunelling
IP routing:
RC obtains forwarding information (VPI/VCI, traffic mode)
L2TP tunelling:
RC obtains forwarding information (VPI/VCI, traffic mode)
If traffic mode is PPPoE over AAL5, forwarding information also includes PPPoE session ID and FP-side MAC DA

Segmentation CP sends and incoming IP packet through the circulation path, building AAL5 PDU
TxByte processor:

strips IP, UDP and L2TP header	**L2TP**
verifies the UDP checksum	**L2TP**
adds PPPoE and Ethernet header	**PPPoE**
adds Ethernet FCS	**PPPoE**

FP generates cells making the given AAL5 PDU, adding the cell header (VPI/VCI) to every cell

Picture 3. Downstream traffic



Picture 4. Host-bound traffic

2. Component design

2.1 Executive processor (XP)

XP performs the following functions:

- Initialization of BMU, TLU, QMU and CP clusters
- Host communication: sending buffers and descriptors between the host and CPs*
- Maintaining the L2TP tunnel timers

- - Not implemented by Motorola.

2.1.1 Timers

The timer is used for monitoring activity on L2TP tunnel. TLU contains a table of timers. Timer ID is an index to this table. The entry of the Timer table contains a 64-bit cycle counter. Every time the DSLAM Uplink Networking Line Card application receives an L2TP data packet, it modifies the corresponding entry in a Timer table, recording the current NPU cycle counter. Working in parallel with CP, XP continuously walks through every entry in the Timer Table. For every entry, it compares the current NPU cycle counter with the cycle counter recorded in the entry. If the difference exceeds the threshold value (also a part of a Timer entry), a message to the host is generated (Message generating code is host-specific and is not implemented).
The purpose of such timer mechanism is to allow host to generate a “Hello” control L2TP packet, according to the rules of RFC 2661. The host sees only L2TP control packets. It never sees L2TP data packets flowing through the DSLAM Uplink Networking Line Card application; therefore it needs some mechanism to monitor this activity.
See the section “GbE cluster. MacRx thread for more details”.

2.2 Fabric Port (FP)

Fabric Port performs reception and transmission of the cells to and from the cell bus.
On reception, two components participate: ***Byte processor*** and ***Descriptor Build Engine (DBE).***
Byte processor operates on a cell header. The cell header contains VPI/VCI (24 bits) and other information, including the EoM (End of PDU) bit. If this bit is set, the cell is the last within a PDU.
Byte processor launches VPI/VCI lookup and selects a target queue, which corresponds to one of CPs in the Reassembly cluster. DBE receives the lookup result and builds the cell descriptor. Cell descriptor (see ***RasCellDescriptor*** description in the QMU section) contains the following information:

- The whole 4-byte cell header, including VPI/VCI and EoM bit
- VC index (information from TLU)
- Processing mode (information from TLU)
- Cell buffer handler

2.3 Reassembly cluster

Reassembly cluster consists of 4 CPs performing the same functionality. Each CP has a dedicated input queue. FP selects a queue to put a new cell into in a round-robin fashion, so every CP gets its share of traffic. Each CP has 2 threads running on a Risc Core – *RasIn* and *RasOut*. The SDP forwarding path is configured for the recirculation mode.
All 4 CPs share a table located in DMEM. The table contains 4096 entries, each of which represents one VC. The entry contains the ***reassembly context,*** information that has to be kept between subsequent cells of the same AAL5 PDU.

2.3.1 Initialization

The DCPmain program, which serves at the entry point for the Reassembly cluster code, is fairly simple. It initializes the various CPI services which receive and transmit processing will rely on (kernel, buffer, pdu, table, and queue services), initializes the buffer pool to be used by this CPRC's receive processing for buffer allocations, initializes SDP recirculation path. Once initialization completes, the main program calls the "*RasIn*" processing which will never return.

2.3.2 CPRC RasIn Thread

The CPRC "**RasIn**" thread consists of a large While(1) loop which repeats the following steps endlessly:

- Waits and receives a cell descriptor from the dedicated input queue
- Looks in the context table to see if there is a reassembly operation in progress. If the entry for that VC is empty, than the incoming cell is considered to be the first in the PDU. The new buffer is allocated for that PDU and its handle is stored in the context table, as well as the offset within a PDU. If the entry is not empty, the buffer handle and offset is taken from there
- Stores buffer handle, VC index, processing mode (not used in Reassembly cluster), offset and EOM status (last cell/not last cell) in Merge Space and initiates DMA of the cell into the recirculation path. Offset is a number 0, 1,2…, meaning the position of the cell within a PDU. The byte offset can be calculated by multiplying it by 48. Buffer handle is the one of AAL5 PDU being reassembled.

It is possible that different CPs work on reassembly of the same PDU. Therefore, access to the context table in DMEM is protected with a token.

2.3.3 CPRC RasOut Thread

The CPRC "RasOut" thread consists of a large While (1) loop which repeats the following steps endlessly:

- Waits and receives a PDU handle. The PDU handle corresponds to the cell that just went through the recirculation circle
- CPRC reads the buffer handle, offset, VC index, processing mode and EOM status from the Extract space
- CPRC uses the buffer handle and offset information to set up DMA. Buffer handle is the buffer handle of an entire AAL5 PDU (don't be confused with a PDU handle, which is a CPI abstraction for the processing scope).
- CPRC waits until the DMA is complete.
- If the EOM status =1, the cell was the last cell of a PDU. Therefore, reassembly of this PDU is finished. CPRC builds the PDU descriptor and enqueues it into the Classification cluster queue. AAL5 PDU descriptor contains:
  - Buffer handle
  - PDU length
  - VC index
  - Processing mode

  CPRC takes the PDU length from extract space, because it will contain the AAL5 PDU trailer (see below the description of RxByte).

2.3.4 TxByte

TxByte microcode is simple. First, it reads VC index, processing mode, offset end EOM status from the Merge Space, sends them "downstream". Then it forwards downstream the cell content, reading it from its PayloadIn bus.
Because TxByte output FIFO is connected to the RxByte FIFO via the recirculation mechanism, RxByte will see output stream of TxByte as its own input stream.

2.3.5 RxByte

RxByte receives its input stream from TxByte. First, the input stream contains the cell parameters:

- VC index
- processing mode
- offset
- Buffer handle
- EOM status.

Then, it contains 48 bytes of a cell. RxByte writes parameters into the Extract space. Then it sets L1DONE flag, so the CPRC can get the PDU handle and start processing the cell. Then it forwards 48 bytes of a cell to the DMA engine. At the end RxByte switches the scope. In addition to forwarding to the DMA engine, RxByte stores the last 8 bytes of a cell in Extract Space. For the last cell, these 8 bytes will contain the AAL5 PDU trailer.

2.4 Classification cluster

Classification "cluster" is not a real cluster, because it consists of only 2 CPs. Both CPs of the Classification cluster share a single input queue, where the Reassembly CPs send the AAL5 PDUs.

2.4.1 Initialization

Classification "cluster" doesn't have its own exclusive DCPmain() entry point. It shares the entry point with 2 CPs of Segmentation cluster, because technically they are located within the same NPU cluster. This combined DCPmain entry point performs different operations for Classification and Segmentation CPs. The common operation is to initialize BMU, QMU and other CPI services.
For Classification CP it performs the following functions:

- Initializes TLU lookups for VC table, Session table and IP table
- Configures the SDP recirculation path. It sets the DSLAM internal FP-side Ethernet MAC address into the Control space of RxByte
- Initializes 2-way queue sharing for a Classification cluster Input queue
- Launches 2 Threads: ClassIn and ClassOut

2.4.2 CPRC ClassIn Thread

The CPRC "**ClassIn**" thread consists of a large While(1) loop which repeats the following steps endlessly:

- Waits and receives an AAL5 PDU descriptor from the shared Input queue
- Stores VC index, processing mode, and PDU length in Merge Space and initiates DMA of the PDU into the recirculation path.

2.4.3 CPRC ClassOut Thread

The CPRC "ClassOut" thread performs most of the decision making about the AAL5 PDU.
The ClassOut thread consists of a large While(1) loop, which is passed once for every received PDU. First, ClassOut thread blocks waiting on a PDU handle. When the PDU handle is obtained, CPRC reads from the extract space: processing mode, PDU length, BytesToSkip and PDU status. Further processing depends on a processing mode. BytesToSkip is parameter of PDU calculated by the RxByte. It means how many bytes have to be skipped before the actual payload. BytesToSkip depends on processing mode only. For IP over AAL5 PDU Header length may include only LLC-header length. For PPPoE L2TP, Header length may include LLC-header length, the size of Ethernet and PPPoE headers
Depending on a processing mode, ClassOut thread selects on of the 3 possible major paths: IP routing, L2TP tunneling/PPP or L2TP tunneling/PPPoE

2.4.3.1 IP routing

The CPRC obtains results of the IP DA Lookup launched by the RxByte. There could be 4 possible results of the lookup:

- Forwarding information present and valid: MAC DA, and the output port is GbE
- IP DA = DSLAM Internal addr
- Forwarding information not present or invalid
- Forwarding information is incomplete. MAC DA not present, but it is known that the route is valid and the output port is GbE

In the first case, PDU will be forwarded to GbE cluster
In the third case, PDU will be discarded
In the second and forth case, PDU will be forwarded to the host

2.4.3.2 L2TP tunneling/PPP

The CPRC obtains results of the VC index lookup launched by the RxByte. There could be 2 possible results of the lookup:

- Forwarding information present: MAC DA, IP DA, UDP dest port, L2TP tunnel and session ID
- If L2TP session ID =0, L2TP session has not been established yet, because 0 is reserved number

In the first case, PDU will be forwarded to GbE cluster
In the second case, CPRC looks onto the 2-byte protocol ID of the PPP frame. RxByte extracts and puts this ID into the extract space. If the protocol is the **control** protocol (bit 15 of the protocol ID is not 0), PDU will be forwarded to the host, otherwise it is discarded.

2.4.3.3 L2TP tunneling/PPPoE

The CPRC obtains results of the PPPoE session ID lookup launched by the RxByte. There could be 2 possible results of the lookup:

- Forwarding information present: MAC DA, IP DA, UDP dest port, L2TP tunnel and session ID
- Forwarding information not present: lookup was not successful.

In the first case, PDU will be forwarded to GbE cluster
In the second case, CPRC looks onto the 2-byte protocol ID of the PPP frame. RxByte extracts and puts this ID into the extract space. If the protocol is the **control** protocol (bit 15 of the protocol ID is not 0), PDU will be forwarded to the host, otherwise it is discarded.

2.4.3.4 Special cases

If the PDU is PPPoE discovery frame, RxByte determines that its destination is host, and sets the status in the Extract space accordingly. In this case no PPPoE session lookup is launched, and therefore no result is obtained.
Also RxByte can determine that PDU is not valid. See below the detailed description of the RxByte. In this case the status in the Extract space will show that the PDU should be discarded.

2.4.3.5 CRC verification

After CPRC has classified the PDU and determined its destination, it waits for the DMA to end. At the end of the PDU, RxByte puts into the extract space the CRC verification status. CPRC reads the status and if the CRC verification had failed, it discards the PDU instead of sending it to the GbE cluster or host.

2.4.3.6 Building a descriptor and forwarding PDU

If CPRC had determined that the destination is host, it builds the following descriptor:

- Buffer handle
- Payload type: AAL5
- AAL5 PDU size
- VC index

Then it forwards it to the XP-owned queue. XP is the ultimate middlemen for CP-host communication
If CPRC had determined that the destination is GbE, it builds the following descriptor:

- Buffer handle
- MAC client: macClientTxIp or macClientTxL2TP
- BytesToSkip
- Payload length. Includes only payload bytes that will be sent out by GbE cluster, without any AAL5-specific or PPPoE specific headers, trailers or pads
- MAC DA
- IP DA (L2TP only)
- UDP dest Port (L2TP only)
- L2TP tunnel ID
- L2TP session ID
- Payload checksum. RxByte calculates it and puts into the extract space. Used only with L2TP

Then it forwards it to the GbE-owned queue.

If RxByte or CPRC have determined that PDU should be discarded, the buffer handle is freed and no descriptor is forwarded.

2.4.4 TxByte

TxByte microcode is simple. First, it reads VC index, processing mode, and payload length from the Merge Space, sends them “downstream”. Then it forwards downstream the PDU content, reading it from its PayloadIn bus.
Because TxByte output FIFO is connected to the RxByte FIFO via the recirculation mechanism, RxByte will see output stream of TxByte as its own input stream.

2.4.5 RxByte

RxByte microcode first reads the parameters: VC index, processing mode and payload length from the input stream. It saves the in the extract space. Then the processing depends on a processing mode.
Processing mode is actually a bit field, specifying various characteristics of a VC:

- Forwarding mode: IP routing, PPP – L2TP or PPPoE – L2TP
- LLC-encapsulation or VC-multiplexing
- For PPPoE, whether Ethernet FCS is present or not

bits 15-8: currently unused
bit 7: How to forward out of GbE:
0 = bridge (Protocol over AAL5 = 0) or route (Protocol over AAL5 = 1, 2, 3, 4)
1 = L2TP (Protocol over AAL5 = 1 or 4)

bits 6-4: Protocol over AAL5: Enet, PPPoE, IPoE, IP or PPP
0 = Ethernet
1 = PPP over Ethernet
2 = IP over Ethernet
3 = IP
4 = PPP

bit 3: LLC or VC mux
0 = VC mux (No LLC header)
1 = LLC/SNAP header

bit 2: Whether the Enet over AAL5 frame contains FCS
0 = no FCS
1 = FCS present

The main idea is that the actual PDU must correspond to its processing mode. Here is the table that lists all supported values for the processing mode, with the corresponding LLC headers.

Table 1. Processing modes

Mode	Value, Hexadecimal	Value, Binary	LLC header
IP routing, LLC	38	00111000	AA AA 03 00 00 00 08 00
IP routing, VC-mux	30	00110000	none
PPPoE LLC, enet FCS	9C	10011100	AA AA 03 00 80 C2 00 01
PPPoE VC-mux, FCS	94	10010100	none
PPPoE VC-mux, no FCS	90	10010000	none
PPPoE LLC, no enet FCS	98	10011000	AA AA 03 00 80 C2 00 07
PPP, LLC	C8	11001000	FE FE 03 CF
PPP, VC-mux	30	00110000	none

RxByte selects the path through the microcode according to the processing mode.
First, it checks PDU for LLC-header validity. If LLC header is not valid, it sets the marks this PDU as bad in the Extract Space (using the *Status* field)
Then, processing depends on a mode. The common thing among all the paths within RxByte is that the whole AAL5 PDU is preserved. Classification cluster doesn't strip anything from the AAL5 PDU. Every byte of the AAL5 PDU is sent to the DMA engine
Another common thing is CRC verification. Every incoming byte of the AAL5 PDU goes into the CRC-32 block. At the end of the PDU, RxByte verifies the CRC validity by comparing the accumulated CRC value to the "magic number". If CRC is not valid, it sets the corresponding bit of the *Status* field in the Extract space.
One more common thing is calculation of the "BytesToSkip" value. For each PDU, RxByte maintains a counter. This counter is initialized to 0 at the beginning of the PDU. Every time RxByte processes the byte which is not a part of payload, but a part of the external header (like LLC-header, Ethernet header or PPPoE header), counter is incremented.

2.4.5.1 IP routing

RxByte launches the IP DA lookup and then sets the L1DONE flag. At the end of the PDU it checks the CRC validity

2.4.5.2 PPP over Ethernet

RxByte verifies the Ethernet and PPPoE headers. The following conditions should be satisfied for the PDU to be considered valid:

- MAC DA = DSLAM internal FP-side MAC address **OR** MAC DA is broadcast
- MAC type = 8863 or 8864 (PPPoE discovery or session)
- First byte of PPPoE header = 0x11
- If the MAC type = 8864 (PPPoE session), MAC address must be DSLAM internal address (no broadcast) and the second byte of PPPoE header must be 0

If the frame is valid PPPoE discovery frame, it is marked in the Extract space as such
If the frame is not valid, it is marked in the Extract space as such, with the corresponding error code.
If the frame is valid PPPoE session frame, the PPPoE session lookup is launched. PPPoE session ID is a part of PPPoE header. RxByte saves the payload length in the internal register and Extract space.
Then the payload is streamed to the DMA. RxByte is able to determine how long the payload is and distinguish it from AAL5 padding, Ethernet FCS or other extra bytes at the end of the PDU, because payload length is a part of PPPoE header. The extra bytes are not added to the payload checksum (see below).

2.4.5.3 PPP

RxByte launches VC-index lookup. In case of PPP, payload length is equal to AAL5 payload length. RxByte knows the AAL5 payload length, because the TxByte as a part of the “pre-PDU” parameters sent the AAL5 payload length to it.

2.4.5.4 Common for PPP and PPP over Ethernet

RxByte saves the PPP protocol ID in the Extract space. Then it sets the L1DONE flag.
RxByte calculates the **Payload checksum**. It includes all the bytes, which are part of PPP frame, including the protocol ID, but NOT including PPPoE and other external headers. Payload checksum is 16-bit checksum. It is needed to transmit the UDP checksum with L2TP packet out of the GbE port.

2.5 GbE cluster

Gbe cluster here serves both transmit and receive sides of GbE port.

2.5.1 Initialization

The DCPmain program, which serves at the entry point for the Reassembly cluster code, performs the following functions:

- Initializes CPI services
- Configures SDP. GbE internal MAC address is set in the control spaces of both RxByte and TxByte
- Initializes IP DA TLU lookup
- Launches 2 threads: MacRx to serve reception and MacTx to serve transmission

2.5.2 CPRC MacRx thread

The CPRC “**MacRx**” thread consists of a large While(1) loop which repeats the following steps endlessly:

- Waits and receives the PDU handle from the RxByte
- Checks the type (forwarding path) of the incoming PDU. It can be IP packet or ARP request. Everything else is considered to be invalid and discarded. If the frame is ARP request, it is forwarded to the host. If it is IP packet, CPRC checks if the IP DA = DSLAM internal IP address. IP DA is taken from the Extract space. If IP DA is not the internal address of the DSLAM, then CPRC assumes that the packet is going to be routed. Otherwise the packet is either L2TP or host-bound.

  #### 2.5.2.1 IP routing

  CPRC obtains the IP DA lookup result. If the route (forwarding information) is present, CPRC builds the IP packet descriptor and forwards it to the Segmentation cluster. The IP packet descriptor contains:
  - Buffer handle
  - Packet size
  - VPI/VCI – from TLU IP routing table entry
  - Processing mode - from TLU IP routing table entry

  Processing mode is the same bit field format as the one used in the classification cluster. It must contain bits specifying the IP routing as a forwarding type. It may define LLC-encapsulation or VC-multiplexing.
  There are some cases when MacRx will generate ICMP message with corresponding descriptor and send it to the GbE Tx side. These are:
  - Route is not present. In this case the ICMP message “Destination unreachable” is generated.
  - Route is located on GbE side. In this case the ICMP message “Redirect” is generated. Both the ICMP message and the original IP packet are forwarded out of GbE port
  - TTL expired. TTL is a part of IP packet header and is located in Extract Space. In this case the IP packet is not forwarded and the “Time Exceeded” ICMP message is generated.

2.5.2.2 L2TP tunneling

If IP DA is the internal address of the DSLAM, CPRC reads the first 64 bytes of the packet from the SDRAM to DMEM. It then checks the following condition to determine if the processing type is indeed L2TP tunneling:

- The protocol must be UDP - 17. Otherwise the packet is forwarded to the host.
- UDP dest port must be the UDP port assigned for the L2TP – 1701. Otherwise the packet is forwarded to the host
- UDP datagram length should match the IP packet length. Otherwise the packet is discarded
- L2TP packet type must be ***data,*** not ***control***. Otherwise the packet is forwarded to the host
- L2TP version must be 2. Otherwise the packet is discarded.
- L2TP packet length should match the IP packet length. Otherwise the packet is discarded.
- L2TP tunnel ID must be not 0. Otherwise the packet is forwarded to the host
- L2TP session ID must be not 0. Otherwise the packet is forwarded to the host

Then, if the packet is valid L2TP data packet, MacRx thread launches the L2TP session lookup.
If the lookup returns L2TP forwarding information successfully:

- L2TP forwarding information is used to build a descriptor for the Segmentation cluster
- L2TP forwarding information contains Timer ID. Each L2TP tunnel has a timer associated with it. The timer is used for monitoring activity on L2TP tunnel. TLU contains a table of timers. Timer ID is an index to this table. The entry of the Timer table contains a 64-bit cycle counter. Every time the DSLAM application receives an L2TP data packet, it modifies the corresponding entry in a Timer table, recording the current NPU cycle counter. Working in parallel with CP, XP continuously walks through every entry in the Timer Table. For every entry, it compares the current NPU cycle counter with the cycle counter recorded in the entry. If the difference exceeds the threshold value (also a part of a Timer entry), a message to the host is generated (Message generating code is host-specific and is not implemented).
  The purpose of such timer mechanism is to allow host to generate a “Hello” control L2TP packet, according to the rules of RFC 2661. The host sees only L2TP control packets. It never sees L2TP data packets flowing through the DSLAM application, therefore it needs some mechanism to monitor this activity
  So, the MacRx thread sends a TLU request “Modify Entry” for every received data L2TP packet, using the Timer ID as an index and recording the current NPU cycle counter.

The L2TP forwarding information (FP direction) contains the following entries:

- VPI/VCI
- Processing mode (PPP or PPPoE, LLC-encapsulation or VC-multiplexing and so on)
- PPPoE session (PPP over Ethernet mode only)
- FP-side MAC DA (PPP over Ethernet mode only)

2.5.3 CPRC MacTx thread

CPRC **MacTx** thread initiates forwarding IP packets, L2TP packets or Ethernet frames to the GbE port
The CPRC “**MacTx**” thread consists of a large While(1) loop which repeats the following steps endlessly:

- Waits and dequeues a descriptor from the shared GbE cluster queue. A single input queue is shared between 4 CPs of the GbE cluster. A descriptor contains the following information:
  - Buffer handle
  - ***MAC client ID*** (what kind of processing is needed): Ethernet, IP or L2TP
  - Bytes to skip before payload. The buffer may begin with bytes that are not part of payload and should not be transmitted
  - Payload length. The buffer may contain extra bytes after the payload, which should not be transmitted as well.
  - IP forwarding information: MAC DA
  - L2TP forwarding information: UDP destination port, L2TP tunnel ID and session ID, payload checksum

The descriptor may come from 3 sources:
- Classification cluster. This is the main upstream data flow. It can define IP packet or L2TP packet
- Host. Host may want to transmit IP packets and Ethernet frames out of GbE port. If the host wants to transmit L2TP packet, it actually builds the whole IP packet or Ethernet frame, since L2TP packet is always encapsulated within IP packet
- Gbe MacRx thread. These are ICMP packets and redirected IP packets

- Prepares the Merge Space for TxByte SDP. Merge space will contain all the needed information from the descriptor, including the **MAC client ID**. For L2TP packets, Merge Space will contain IP SA and DA, UDP source and destination ports, L2TP tunnel ID and session ID, IP packet length, UDP datagram length and L2TP packet length
- Launches the DMA of the buffer from SDRAM to the TxByte SDP

2.5.4 TxByte SDP

TxByte SDP builds and transmits IP packets, Ethernet frames and L2TP packets out of GbE port. TxByte selects the processing path according to the MAC client ID value, which it takes from its Merge Space.

2.5.4.1 MAC client Ethernet

In this case the input stream of TxByte contains the whole Ethernet frame without FCS. TxByte sends the input bytes downstream, adding the FCS at the end. FCS is CRC-32, calculated using the CRC block

2.5.4.2 MAC client IP

In this case the input stream of TxByte contains the IP packet. TxByte performs the following actions:
- Adds the Ethernet header, containing MAC SA, MAC DA and protocol (IP). MAC SA and MAC DA are taken from the Merge Space. Sends the Ethernet header downstream
- Decrements the TTL value of IP packet header, changes the IP checksum accordingly. Sends the modified IP packet bytes downstream
- Adds the Ethernet FCS

2.5.4.3 MAC client L2TP

In this case the input stream of TxByte contains the AAL5 PDU, which was built by the Reassembly cluster. TxByte performs the following actions:
- Skips the header bytes. The number of bytes to skip is determined by the parameter “BytesToSkip”, which is taken from the Merge Space
- Adds the Ethernet header, containing MAC SA, MAC DA and protocol (IP). MAC SA and MAC DA are taken from the Merge Space. Sends the Ethernet header downstream
- Builds and adds the IP header. IP SA, IP DA, IP packet length are taken from the Merge Space.
- Builds and adds the UDP header. UDP checksum is calculated using the Payload checksum value taken from the Merge Space. UDP source port, UDP destination port, UDP datagram length are also taken from the Merge Space
- Builds and adds the L2TP header. L2TP tunnel ID and session ID, L2TP packet lengths are taken from the Merge Space.
- Sends the payload bytes downstream
- Adds the Ethernet FCS

2.6 Segmentation cluster

Segmentation cluster doesn’t actually build the segments, or ATM cells. It merely builds the AAL5 PDU. Segmentaion cluster is not a real cluster either. It contains just 2 CPs, which are technically located within the same cluster as Classification CPs

2.6.1 Initalization

Segmentaion “cluster” doesn’t have its own exclusive DCPmain() entry point. It shares the entry point with 2 CPs of Classification cluster, because technically they are located within the same NPU cluster.

This combined DCPmain entry point performs different operations for Classification and Segmentation CPs. The common operation is to initialize BMU, QMU and other CPI services.
For Segmentation CP it performs the following functions:

- Configures the SDP recirculation path. It sets the **DSLAM internal FP-side Ethernet MAC address** into the Control space of TxByte
- Initializes 2-way queue sharing for a Segmentation cluster Input queue
- Launches 2 Threads: SegIn and SegOut

2.6.2 CPRC SegIn thread

The CPRC "**SegIn**" thread consists of a large While(1) loop which repeats the following steps endlessly:

- Waits and receives a Segmentation descriptor from the shared Input queue
- Stores VPI/VCI, processing mode, payload length, MAC DA and PPPoE session ID (PPPoE L2TP only) in Merge Space and initiates DMA of the PDU into the recirculation path.

2.6.3 CPRC SegOut Thread

The SegOut thread consists of a large While(1) loop, which is passed once for every received PDU. SegOut thread performs the following actions:

- Waits and obtains a PDU from the RxByte
- Waits for payload to be fully transferred into the SDRAM
- For L2TP packets, checks the UDP checksum verification result. RxByte performs this verification and puts its result into the Extract Space. If UDP checksum is not valid, PDU is discarded
- Selects the input queue for FP. Each CP has two FP queues assigned to it. Each CP "flip-flops" between them for every PDU: the first PDU goes into the first queue, the next one goes into the second, and then the next goes into the first one, and so on. Overall, FP Tx Engine has 4 queues feeding it, 2 queues from each CP
- Builds the descriptor for FP and enqueues it into the selected queue. Descriptor for FP has the following fields:
  - VPI/VCI
  - PDU length

2.6.4 TxByte SDP

TxByte SDP builds an entire AAL5 PDU (without 4 final CRC bytes) and forwards it further into the recirculation path. The way to build the AAL5 PDU depends on a processing mode. First, the TxByte SDP reads the processing mode, VPI/VCI and PDU length from Merge Space. Then it forwards the processing mode and VPI/VCI further downstream, so RxByte can see them just before the actual PDU
If the mode specifies LLC-encapsulation, TxByte generates the LLC header according to the forwarding mode. See the table 1 above.
Then, if the mode specifies IP routing, TxByte simply forwards payload bytes downstream
If the mode specifies PPP over AAL5 or PPPoE over AAL5, TxByte does the following:

- Skips IP header, UDP header and L2TP header. Saves the UDP checksum in the internal register.
- Starts calculating the UDP checksum. UDP checksum is the combination of payload checksum, L2TP and UDP header checksum and checksum of several fields within IP header (so called "pseudoheader").

Now, if the mode specifies PPP over Ethernet, TxByte builds and sends downstream the Ethernet header and PPPoE header. MAC SA, MAC DA, PPPoE session ID are taken from the Merge Space. Otherwise, if the mode specifies PPP over AAL5, TxByte goes right to the forwarding the payload bytes

- Forwards payload bytes downstream, adding the bytes to the UDP checksum on a way
- For PPPoE only: if the resulting Ethernet frame length is less than 64 bytes (60 bytes in case Ethernet FCS should be added), TxByte sends downstream the corresponding number of padding zeros
- For PPPoE only: if the processing mode specifies that the Ethernet FCS header should be added, TxByte reads the 4-byte value from the CRC block and sends downstream. For PPPoE, every byte belonging to the Ethernet frame (including Ethernet header, PPPoE header and payload) goes through the CRC-32 block

Now, for all the processing modes, the AAL5 PDU padding is added, and then the AAL5 PDU trailer.
The only thing that is not added here is AAL5 PDU CRC. The RxByte does it.
Finally, if the processing mode is PPP or PPP over AAL5, TxByte verifies the UDP checksum. It compares the calculated value of UDP checksum with the stored value, which was taken from the actual UDP header. If they match, TxByte sends dowstream the value 1. Otherwise, it sends the value 0.

2.6.5 RxByte SDP

RxByte SDP receives PDU from the TxByte, because they are connected via the recirculation path
First, RxByte SDP receives VPI/VCI and processing mode, and stores them in Extract space.
Then, it receives the AAL5 PDU and sends it upstream, to the DMA Engine. Every byte is also sent to the CRC-32 block.
Upon reaching the end of stream, RxByte reads the CRC value from the CRC-32 block and sends it upstream.
Finally, if the processing mode is PPP or PPPoE, it writes the UDP checksum verification result (0 or 1) into the Extract Space.

2.7 TLU

There are the following tables in TLU: VC map table, VC table, IP routing table, PPP session table and Timer table

2.7.1 VC map table

VC map table is used by FP to convert 24-bit VPI/VCI into the continuous 12-bit VC index. It is a Hash-Tree-Key table. Entry size is 8 bytes. Format of entry:

Table 2. VC map table

Field name	**Size, bytes**	**Purpose**
Mode	2	Processing mode
Unused	2	
VcIndex	4	VC index

VC map table key is 32-bit wide. It contains VPI/VCI at the same place as 4-byte cell header:

4 zero bits	8-bit VPI	16-bit VCI	4 zero bits

2.7.2 VC table

The Classification cluster uses VC table to obtain the L2TP forwarding information for PPP over AAL5 frames. VC table is simple data table. The key to this table is VC index. Entry size is 64 bytes. It is mode than needed, to reserve some space for future protocol additions. Format of entry:

Table 3. VC table

Field name	**Size, bytes**	**Purpose**
CellHeader	4	Contains VPI/VCI
Phy	1	
Unused	3	
MAC DA	6	Destination MAC address on GbE port
DestUDP	2	Destination UDP port
DestTunnelID	2	Destination L2TP tunnel ID
DestSessionID	2	Destination L2TP session ID
DestIP	4	Destination IP address

As required by the TLU architecture, the key is 32-bit wide

2.7.3 IP route table

The IP route table is used throughout the DSLAM application to obtain the IP route in case IP routing is performed. On C5 implementation, IP route table is VP Trie-Index-Data table. On C5e/C3e implementation, IP route table is a PFX table. The entry size is 16 bytes. Format of entry:

Table 4. IP route table

Field name	Size, bytes	Purpose
mask	1	IP subnet mask
bitFields	1	***Bitfield:*** Bit 7: 0 - egress port FP 1 - egress port GbE Bit 0: phy channel (used only if bit 7 == 0) Bit 1: incomplete path - no defined mac DA yet Bit 2: 1 - destination host
mode	2	Processing mode
cellHeader	4	VPI/VCI is a part of it
MAC DA	6	Destination Ethernet MAC address
PPoESessionID	2	PPPoE session ID, used only if egress port is FP

The key is 32-bit IP address

2.7.4 PPP session table

The PPP session table contains forwarding information for two different sets of session Ids. One set is the set of PPP over Ethernet session Ids, and the Classification cluster launches the PPPoE session ID lookups. Another set is the set of L2TP session Ids, and the GbE cluster launches theL2TP session ID lookups. PPP session table is a simple data table. The session ID itself is 12-bit value. The key to the PPP session table is 32-bit wide. Bit 14 is used to distinguish between one set or another. For example, PPPoE session Ids 1, 2… are converted to keys 0x00000001, 0x00000002, but the L2TP session Ids 1, 2,… are converted to keys 0x00004001, 0x00004002, … etc.

The entry size is 32 bytes. Format of entry depends on a set of keys. For PPPoE set of keys, format is the following:

Table 5. PPP session table, PPPoE sessions

Field name	Size, bytes	Purpose
Mode	2	Processing mode
BitFields	1	Bit 7: 0 - egress port ATM 1 - egress port GbE Bit 0: phy channel (used only if bit 7 == 0)
Unused	5	
MAC DA	6	Destination MAC address on GbE port
destUDP	2	Destination UDP port
destTunnelID	2	Destination L2TP tunnel ID
destSessionID	2	Destination L2TP session ID
destIP	4	Destination IP address

For L2TP set of keys, format is the following:

Table 6. PPP session table, L2TP sessions

Field name	Size, bytes	Purpose
Mode	2	Processing mode
BitFields	1	Bit 7: 0 - egress port ATM 1 - egress port GbE Bit 0: phy channel (used only if bit 7 == 0)
MAC DA	6	FP-side Ethernet MAC DA (PPP over Ethernet only)
PPPoESesssionID	2	PPP over Ethernet session ID
CellHeader	4	VPI/VCI is a part of it
Timer ID	2	Timer ID corresponding to the L2TP session, which is lookup launched upon; there is one timer per tunnel, so for all sessions within the same tunnel, timerID value is the same

2.7.5 Timer Table

Timer table is used to monitor activity on a L2TP tunnel. Each L2TP tunnel has a corresponding entry in this table. The table is accessed from 2 places within NPU. First, the GbE cluster writes the current cycle counter into the entry every time the L2TP data packet has been received on that tunnel. Second, XP periodically checks every entry in the table, in order to detect tunnels where no activity happened for more than a certain period of time.
Timer table is a simple data table. The entry size is 32 bytes. The key is TimerID and it is a 32-bit wide index into this table.
Format of entry:

Table 7. Timer table

Field name	Size, bytes	Purpose
initialCount	8	Value written by the GbE CP every time a new packet is received
timeToLive	8	Maximum inactivity period. If XP detects that the difference between the *initialCount* and the current cycle counter is more than *timeToLive*, message to the host should be generated.
timerType	2	Value 0 means that the entry is not in use Value 1 means that the timer is in use an linked to the L2TP tunnel
tunnelId	2	An associated tunnel ID

2.8 QMU

QMU Descriptor size is 32 bytes. Descriptor formats:

RasCellDescriptor: sent from FP to the Reassembly cluster

Table 8. RasCellDesctiptor

Type	Name	Size, bytes	Purpose
BsBufHandle	BufHandle	4	Buffer handle for a cell
Int32u	CellHeader	4	The ATM header of a cell
Int16u	Mode	2	Processing mode
Int8u	Phy	1	Phy address on a FP
Int8u	Unused	1	
Int32u	VcIndex	4	VC index
Int32u	Unused	16	

RasToClassDesctiptor: sent from Reassembly cluster to Classification cluster:

Table 9. RasToClassDescriptor

Type	Name	Size, bytes	Purpose
BsBufHandle	BufHandle	4	Buffer handle for an AAL5 PDU
Int16u	Mode	2	Processing mode
Int32u	VcIndex	2	VC index
Int16u	pduLength	2	Length of AAL5 PDU without trailer and padding. Taken from the AAL5 PDU trailer
Int16u	bufLength	2	Length of the whole buffer, including the padding and trailer
Int8u	unused	20	

ClassToGbeDescriptor. This descriptor is sent to the GbE cluster from 3 different sources: Classification cluster, GbE cluster itself (ICMP packets) and host.

Table 10. ClassToGbeDescriptor

Type	Name	Size, bytes	Purpose
BsBufHandle	BufHandle	4	Buffer handle for an AAL5 PDU
Int8u	macClient	1	Defines the processing path through the TxByte
Int8u	BytesToSkip	1	A number of bytes in the buffer before the payload starts
Int16u	PayloadLength	2	Payload length
Int16u + Int32u	macDA	6	Destination MAC address
Int16u	Vid	2	Reserved; must be 0xFFFF
Int16u	TunnelID	2	L2TP tunnel ID

Int16u	SessionID	2	L2TP session ID
Int32u	destIP	4	Destination IP address
Int16u	DestUDP	2	Destination UDP port
Int16u	payloadChecksum	2	Payload checksum

GbeToSegDescriptor: This descriptor is sent from the Gigabit cluster to the Segmentation cluster

Table 11. GbeToSegDescriptor

Type	Name	Size, bytes	Purpose
BufHandle	BufHandle	4	Buffer handle for the IP packet
Int8u	PayloadType	1	Constant value PAYLOAD_TYPE_IP
Int8u	Unused		
Int16u	payloadLength	2	Length of payload within the buffer For IP packets – length of the whole IP packet For L2TP packets – length of the L2TP payload
Int16u	Mode	2	Processing mode
Int32u	CellHeader	4	Contains VPI/VCI
Int32u + Int16u	MAC DA	6	FP-side Ethernet MAC DA (used with PPPoE)
Int16u	PPPoESessionID	2	PPPoE session ID
Int8u	Phy	1	Phy channel on FP
Int8u	Unused	11	

SegFwdDescriptor: This descriptor is sent from the Segmentation cluster to the FP:

Table 12. SegFwdDescriptor

Type	Name	Size, bytes	Purpose
BsBufHandle	BufHandle	4	Buffer handle for the AAL5 PDU
Int16u	Length	2	AAL5 PDU length (The whole buffer)
Int8u	Phy	1	Phy channel on a FP
Int8u	unused	1	
Int32u	CellHeader	4	Contains VPI/VCI
Int32u	Reserved	20	

HostDescriptor: This descriptor is sent to the host. Format of the descriptor depends on the type of packet or frame, which is being sent. The host uses payloadType value to determine the type

HostDescriptor for AAL5 PDU (sent by the Classification cluster):

Table 13. HostDescriptor, PAYLOAD_TYPE_AAL5

Type	Name	Size, bytes	Purpose
BsBufHandle	bufHandle	4	Buffer handle for the PDU
Int8u	payloadType	1	For this descriptor: PAYLOAD_TYPE_AAL5
Int8u	Unused	1	
Int16u	payloadSize	2	Size of the PDU
Int16u	vcIndex	2	Index of VC where the PDU came from
Int8u	unused	22	

HostDescriptor for IP packet (sent by the GbE cluster):

Table 14. HostDescriptor, PAYLOAD_TYPE_IP

Type	Name	Size, bytes	Purpose
BsBufHandle	bufHandle	4	Buffer handle for the packet
Int8u	payloadType	1	For this descriptor: PAYLOAD_TYPE_IP
Int8u	Unused	1	
Int16u	payloadSize	2	Size of the packet
Int8u[6]	MAC SA	6	Ethernet MAC SA of the IP packet
Int8u	unused	22	

HostDescriptor for ARP frame (sent by the GbE cluster):

Table 15. HostDescriptor, PAYLOAD_TYPE_ARP

Type	Name	Size, bytes	Purpose
BsBufHandle	BufHandle	4	Buffer handle for the frame (contains the whole Ethernet frame)
Int8u	payloadType	1	For this descriptor: PAYLOAD_TYPE_ARP
Int8u	Unused	1	
Int16u	payloadSize	2	Size of the frame
Int8u[6]	MAC SA	6	Ethernet MAC SA of the frame
Int8u	arpType	1	ARP_BROADCAST or ARP_UNICAST
Int8u	unused	21	

3. List of RFC

- RFC 2661. Layer Two Tunneling Protocol "L2TP"
- RFC 2684 Multiprotocol Encapsulation over ATM Adaptation Layer 5
- RFC 2364 PPP over AAL5
- RFC 2516 A Method for Transmitting PPP Over Ethernet (PPPoE)
- RFC 768 User Datagram Protocol (UDP)