Chapter 4.

System Implementation

4.1. Overview

The “A Day in the Life of a Packet” [32] is referenced as a basic function for our system implementation such as packet transmitting over IQ2000 and resource allocation. Based on the fundamental functions, we can put something like header translation, payload modification and transmit process together with them for forming a whole new function. Figure 4.1 illustrates the fundamental packet flow chart in this system implementation. The detail descriptions of the operation are listed as following.

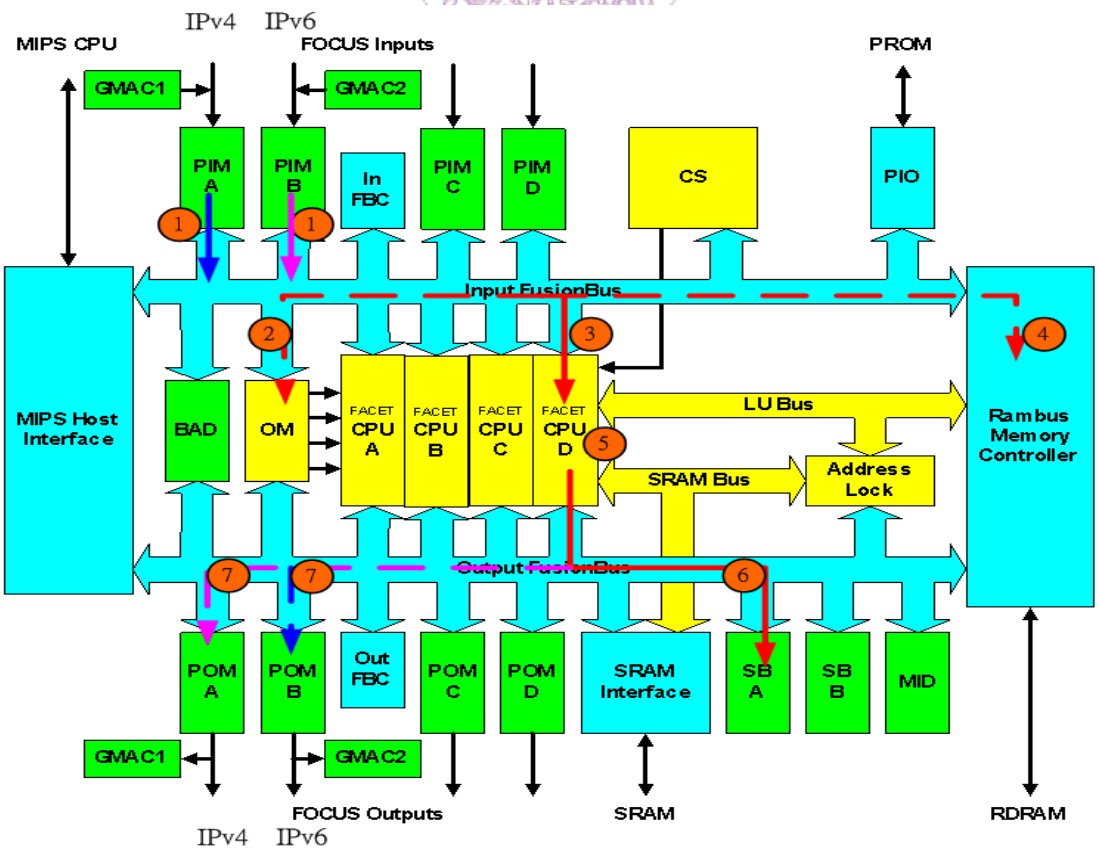


Figure 4.1 Fundamental Packet Flow over Vitesse IQ2000

1. The packets are coming form the FOCUS Gigabit port 0 for IPv4 packet and port 1 for IPv6 packet.
2. The PIM divides the packet either from IPv4 or IPv6 side into “Header” (configurable up 96 Bytes) and “Payload” where the size configuration of the “Header” is important for our implementation in the cases of IPv4-IPv6 and IPv6-IPv4 indicated in the later section.
3. The “Header” part is sent to FACET via OM which will select one of the four FACETs that is available or least busy for process.
4. The “Payload” part is paced in memory over Input Fusion Bus with an indicator within the Input Header Descriptor generated by PIM. Since we will process DNS record translation in “Payload” part, the memory shifting problem described in following sections will be emerged to our system implementation.
5. After the “Payload” completely moves to main memory, if the packet is long enough, the selected FACET will be notified by OM and process the input packet residing in one of the 16 header buffers with various operations like header translation, table lookup, and checksum recalculation and transmission process. In this step, we can put some type of packets to MIPS CPU for special process like DNS packets by placing them in the specified queue in memory.
6. After completing the main operations of our implementation in FACET, it sends the modified “Header” named Output Header including the Output Header Descriptor to Smart Buffer queue waiting for transmitting using the DMA co-processor.
7. The POM fetches the Output Header from the SB queue and combines the modified “Header” and “Payload” in main memory into a complete packet if necessary and forwards the translated packet to port 0 or port 1. Now, the basic data flow over the IQ2000 device is finished.

4.2. FOCUS Cell and Packet Type

The network packets are broken up into data groups called FOCUS Cells or FCells [31] in the period of transferring over FOUCS interface. The numbers of the FCell belonged to a single packet is based on the length of the packet. Each context in the FACET CPU processes one FCell at a time. One FCell could be an entire packet or just the part of the packet according to the length of the packet. Each FCell has an identifier named FCell Header which includes some information about this FCell such as the length and type.

When PIM receives the FCells from the FOCUS interface, it will remove the FCell Headers in front of the FCells and classify the packets into three categories: Header, Complete, and Long, based on the relationship among the Header Size, packet size and FCell length. The definition of the three packet types are listed below [31].

- Header Type

The packet is contained completely within the first FCell. The FCell length is less than or equal to Header Size. We can induce this definition to an inequality as following.

$$Packet\_Length \leq FCell\_Length \leq Header\_Size$$

- Complete Type

The packet is contained within the first FCell but its length is greater than the Header Size.

$$Header\_Size < Packet\_Length < FCell\_Length$$

- Long Type

The packet is not completely contained within the first FCell. That is to say, the packet also has

the “Payload” part which should be moved to main memory.

$$Header\_Size < FCell\_Length < Packet\_Length$$

The packet type is indicated by some information in the Input Descriptor that is referenced by the FACET as the “Header” part comes into the Header Buffer. Something should be concerned is that since the packet which is Long type or Complete type has the extra “Payload” part that must be moved to main memory, the information about what packet type the packet is will be put in the other data structure called Input Complete Message (ICM) which is generated and transferred by PIM over Input Fusion Bus to the location of the top of the correspond Header Buffer in the FACET CPU. For these different packet types, the programmer could define various methods for different processes. For example, the Header type packet usually contains the important information like IPv4 or IPv6 header for necessary translations in our implementation. For Long or Complete type packet, it always has no particular process except for DNS-ALG function and the address translation of IPv6 side ($NS \Leftrightarrow NA$) .We will give a detail description of these data structures in the Header Buffer and the different policies of memory and Buffer allocations for processing the various packets in following sections.

4.3. Buffer Allocation and Memory Management

4.3.1. Header Buffer Allocation

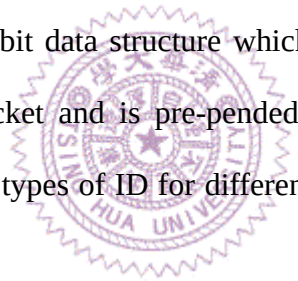
When the IQ2000 receives the external packet, the PIM will separate it into the “Header” and “Payload”, where the “Header” will be sent to the FACET while the “Payload” will be placed in RDRAM. With the respect to the “Header” part in FACET, it will be provisionally placed in one of the 16 Header Buffers which is the temporary packet buffer in FACET. The “Header” includes 2

parts of ID (Input Descriptor) [31] that is generated by PIM for carrying some important information of the packet and IH (Input Header) [31] which is the real header content belonged to the original incoming packet. The Header Buffer will intelligently reserve one proper space for the ID. In fact, there is also one reserved space for OD (Output Descriptor) [31] and ICM (Input Complete Message) [31].

The ICM locates at the top of the Header Buffer, while OD and ID follow it with 4-byte and 2-byte space respectively. Something we should concern is that the OD has 4-byte continued space covering where ID resides in. Following these two data structures is the “Header” part in the rear. We explain these data structures in Header Buffer as below.

- ID (Input Descriptor)

The ID is the format of one 64-bit data structure which has several fields figuring out some important information about this packet and is pre-pended to the header data while the packet comes from the PIM. There are three types of ID for different packet type like Table 4.1, Table 4.2 and Table 4.3 presented.



Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	RESERVED																			E	HLENGTH						
1	CLASS								RESERVED																0	PIMID		0	SRCPORT			

Table 4.1 PIM Input Descriptor of Header Type

Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	BSPC			Data Buffer pointer (DBP)																E	HLENGTH						
1	CLASS								PLENGTH								RESERVED								0	PIMID		0	SRCPORT			

Table 4.2 Input Descriptor of Complete Type

Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	BSPC			Data Buffer pointer (DBP)																0	HLENGTH						
1	CLASS								RESERVED																0	PIMID		0	SRCPORT			

Table 4.3 Input Descriptor of Long Type

Among them, the HLENGTH and PIMID indicate the length of the header data and which PIM device the packet comes from respectively where HLENGTH is decided by the HEADER_SIZE field in PIM Control Register (PIM_CTRL) with maximum value of 96 bytes. The BSPC and DBP in ID of Complete and Long Type specify the location of the “Payload” in RDRAM where BSPC indicates which Data Buffer Space the “payload” is located in and DBP points to which Data Buffer the “Payload” starts. Figure 2.6 in Chapter 2 shows the relationship of BSPC and DBP and how to specify the address of “Payload” in RDRAM.

If the packet has extra “Payload” after the division of PIM, the packet type must be Complete or Long type. Therefore, the payload length is indicated in PLENGTH of the ID of Complete type. We should pay more attention to that since the Long type packet makes the PIM device separate it into more than one FCell, the finish time of data transferring of “Header” and “Payload” to FACET and RDRAM respectively is not consistent. That is, when “Header” part completely gets into Header Buffer, there are still some “Payload” part continuously delivering to RDRAM at this moment. That is why no PLENGTH field in the ID of Long type packet. In IQ2000, it uses the other data structure called Input Complete Message to carry this information of “Payload” length for FACET, just as Table 4.5 presents below.

- OD (Output Descriptor)

OD contains the information of the packet after being modified by FACET. In other words, HLENGTH and PLENGTH identify the “Header” and “Payload” length respectively. Besides, BSPC, DBP, PA, BUFAD are referenced by POM for accessing the “Payload” in RDRAM to form a complete packet and they are left in next subsections for being detail described.

Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TYPE			U1	U2	BSPC			Data Buffer Pointer (DBP) [15:0]																X	HLENGTH							
1	PLENGTH																PA		OFFSET				M	MCID							
2	C	W	DPTI			WMK0			WMK1			WMK2			WMK3		T	TIMESTAMP													
3	BUFAD[31:0]																															

Table 4.4 Output Descriptor

- ICM (Input complete Message)

Word	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	PLENGTH																A	E	S	C	RESERVED				0	PIMID			SRCPORT			
1	0																0	0	0	0	0				0	0		1	0			

Table 4.5 Input Complete Message

4.3.2. Memory Management

The main memory device in IQ2000 is called RDRAM connected by the Rambus Memory Controller. It holds up to 4 GB for system usage, packet buffer and user-defined data structures.

4.3.2.1. Memory usage

In our NAT-PT implementation, we have several data structures that should be constructed in RDRAM to assist the system for proper operation. Each of these data structures has its own functions illustrated as below.

1. Available IPv4 address pool

   Maintaining a number of free IPv4 addresses to substitute for IPv6 nodes who want to access the IPv4 domain.

2. IPv4-IPv6 address mapping table

Maintaining the address mapping between IPv4 address and IPv6 address to make the communication transparent either from IPv4 to IPv6 or vice versa.

3. IPv4-Mac address mapping table

Storing the address mapping between IPv4 address and physical address belonged to the same node in the subnet which is the same with the NAT-PT translator.

4. IPv6-Mac address mapping table

Storing the address mapping between IPv6 address and physical address belonged to the same node in the subnet with the same prefix domain to NAT-PT translator.

We will explain the detail operations with the two address mapping tables as described in 3 and 4 in following sections.

4.3.2.2. Payload Address in RDRAM

IQ2000 defines four different address modes for the payload in RDRAM which indicates how to access it for processing by different methods of address calculation. This function makes the system design more flexible when data in RDRAM must be largely and frequently moved either internal up and down in RDRAM or external to the other devices. Table 4.6 shows how to calculate the address of the payload in RDRAM corresponding to each mode.

In our implementation, we have different address modes for different packet flow that is put together as Table 4.6 shows. Where BSPC identifies which of the eight Data Buffer Spaces contains the first payload DBP, DBP (Data Buffer Pointer) is who points to the first payload Data Buffer, and BUFAD specifies the starting byte address of the payload in RDRAM.

Mode	Payload Address in RDRAM (unit : byte)	Packet Flow
PA=00	(DBP*256) + Buffer Space Base Register selected by BSPC	IPv4→IPv6
PA=01	(DBP*256) + Buffer Space Base Register selected by BSPC + BUFAD	IPv6→IPv4
PA=10	BUFAD	Not used
PA=11	Buffer Space Base Register selected by BSPC + BUFAD	Not used

Table 4.6 Payload Address Mode

4.4. Header Size Problem

For every incoming packet, we allocate one 128-byte Header Buffer for “Header” and one or more Data Buffers for “Payload”. Since the header length of IPv4 and IPv6 are different after and before header translation, the header length will be 20 bytes increased and 20 bytes decreased when IPv4 to IPv6 and IPv6 to IPv4 respectively. Therefore, a proper amount of “Payload” data in RDRAM should be moved to Header Buffer while the “Header” can not contain all of the information which FACET needs to process if the HEADER_SIZE is set too short during IPv6 to IPv4 and a proper amount of “Header” data which is not necessary for process in Header Buffer must be moved to the top of the “Payload” in RDRAM if the HEADER_SIZE is set too long (more than 84) to spark off the inefficiency of the Header Buffer Space during IPv4 to IPv6. However, the configuration of HEADER_SIZE in PIMCTRL is static for all the PIM (PIM A~ PIM D) devices. As a result, a problem of shifting “Payload” data in RDRAM inevitably occurs since we just only have one choice for the configuration of HEADER_SIZE: long or short. But something we should know is that no matter how long the HEADER_SIZE is, we must face the problem that the system performance will go a little down due to the loading of the data movement either from Header Buffer to RDRAM or form RDRAM to Header Buffer and the complexity of implementation for programmers will be a little increased.

We are now discussing the problem of HEADER_SIZE below.

- Case 1: setting a long value to HEADER_SIZE

It seems not to be workable for setting a long value to HEADER_SIZE. As mentioned above, when it is the case of IPv4 to IPv6 with the long value in HEADER_SIZE, FACET should move the additional data in Header Buffer to the top of the “Payload” in RDRAM. However, it is impossible to directly place the data coming from Header Buffer to top of the “Payload” since each “Payload”

has its own Data Buffer Space so that it may overwrite the other one's "Payload" data. Thus, it has to first shift down the whole "Payload" data to bring out one empty space for the Header Buffer data. These operations of the data movement among Header Buffer and RDRAM are shown in Figure 4.2. As presented in Figure 4.2, such data movement operation will produce a lot of system overhead that we think this method is not practicable. So we are forced to adopt the other choice of short configuration to HEADER_SIZE as the policy in our implementation described in Case 2.

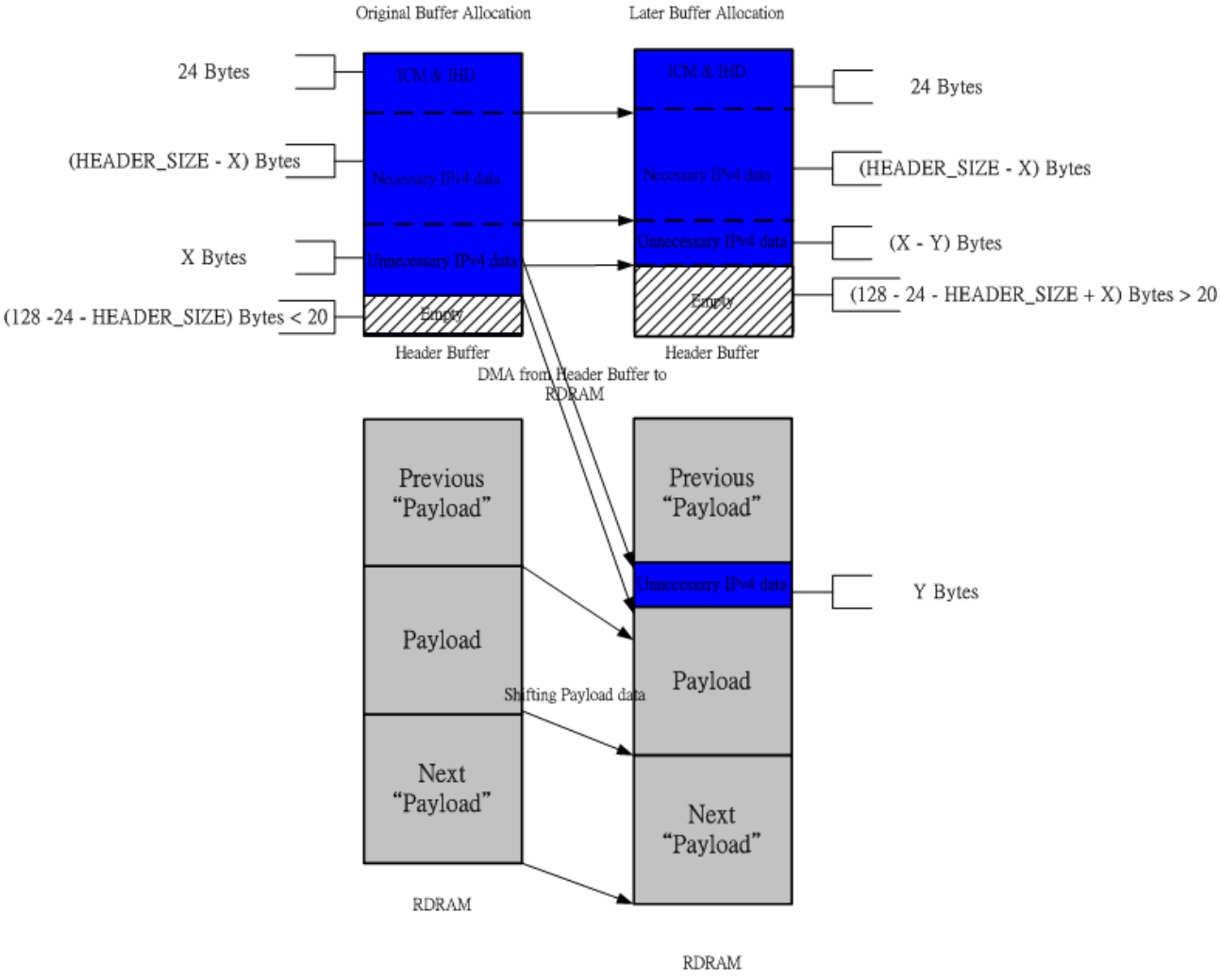


Figure 4.2 Buffer Allocation of Long HEADER_SIZE Configuration

- Case 2: setting a short value to HEADER_SIZE

In the case of IPv6 to IPv4 with a short value to HEADER_SIZE, although we still have the system overhead of moving the necessary data of "Payload" from RDRAM to Header Buffer, we

can use the address mode of payload address mentioned in Chapter 4.3.3.2 to avoid the addition loading of large memory shifting of moving the lower “Payload” data to the higher position which the DBP points to illustrated in Figure 4.3.

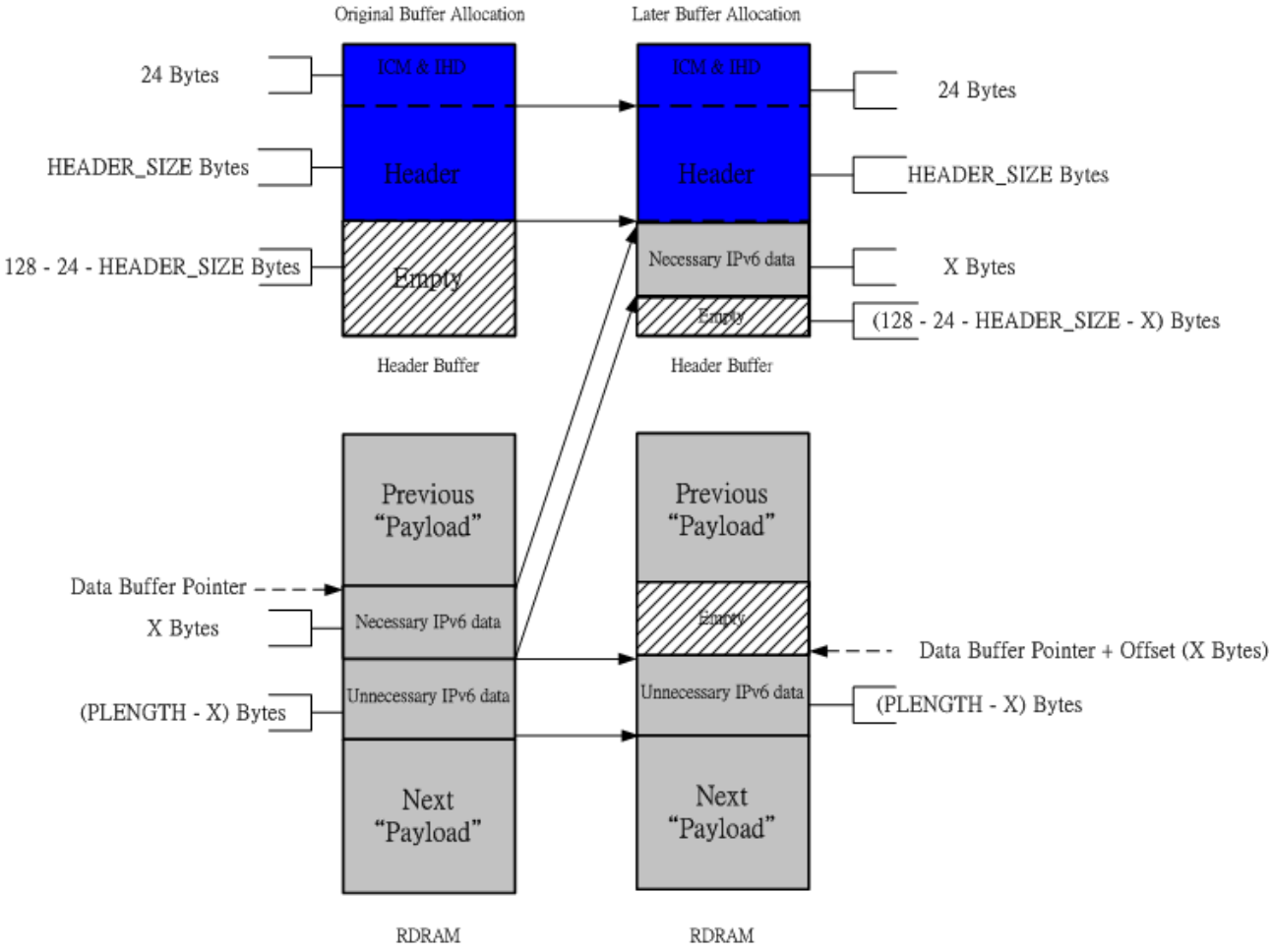


Figure 4.3 Buffer Allocation of Short HEADER_SIZE Configuration

As the determination of giving a short IH to Header Buffer, there is one problem remaining just now that how short the HEADER_SIZE should be configured. Since we have made a decision of choosing the Case 2 to implement our system, the value of HEADER_SIZE must be smaller than the longest length of the incoming IPv6 packets that need to be processed by NAT-PT in evidence. On the other hand, the value of HEADER_SIZE has to be bigger than the longest length of the incoming IPv4 packets that needs to be processed by NAT-PT. Otherwise we will pay lots of additional costs to move some “Payload” data in RDRAM that needs to be coped with by FACET to

Header Buffer. As we know from beforehand observation, the longest IPv4 packet we need to process is TCP [37] packet with a location of byte 76 ($24+52$, 24 bytes for ICM and Packet Descriptor, and 52 bytes for TCP packet from MAC header to checksum field in TCP header) in Header Buffer while it includes the checksum field that needs to be modified at the farthest position among all kinds of incoming IPv4 packets such as ARP, UDP, ICMP, TCP packets. That is, the value of HEADER_SIZE must be bigger than 52.

In our NAT-PT system, we give the value of 64 (bigger than 52 and smaller than 84) to HEADER_SIZE field. For those incoming IPv4 packets from PIM A, the Header Buffer has at least 44-byte space to accommodate those increasing headers after being translated by FACET without undesirable DMA operations of moving extra data to memory since there are only 20-byte IPv6 header increasing after translation from IPv4 to IPv6. Moreover, for those incoming IPv6 packets from PIM B, we assign the policy that uniformly moves 32-byte “Payload” data from memory to Header Buffer since the longest IPv6 packet we need to process is NS (Neighbor Solicitation) [5] packet with a location of byte 96 ($24+72$, 24 bytes for ICM and Packet Descriptor, and 72 bytes for NS packet from MAC header to the “Sender’s Link Address” field in NS packet) in Header Buffer after DMA operation of 32-byte “Payload” data movement from memory. Figure 4.4 and Figure 4.5 shows the variation in Case of IPv4 to IPv6 and IPv6 to IPv4 in Header Buffer with a value of 64 to HEADER_SIZE respectively.

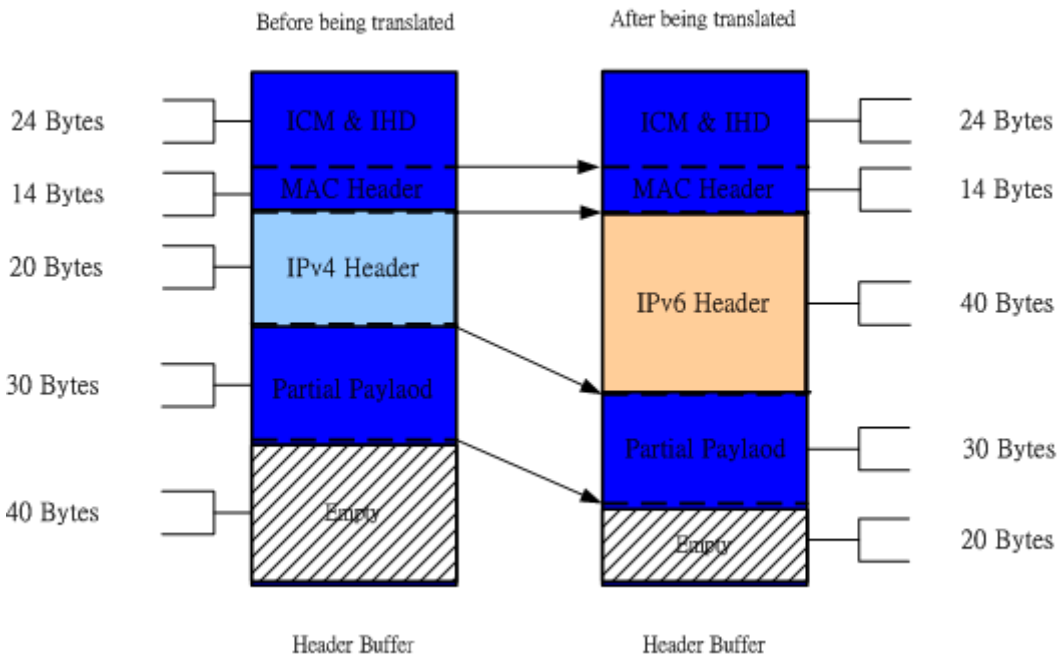


Figure 4.4 Variation of Header Buffer in Case of IPv4 to IPv6

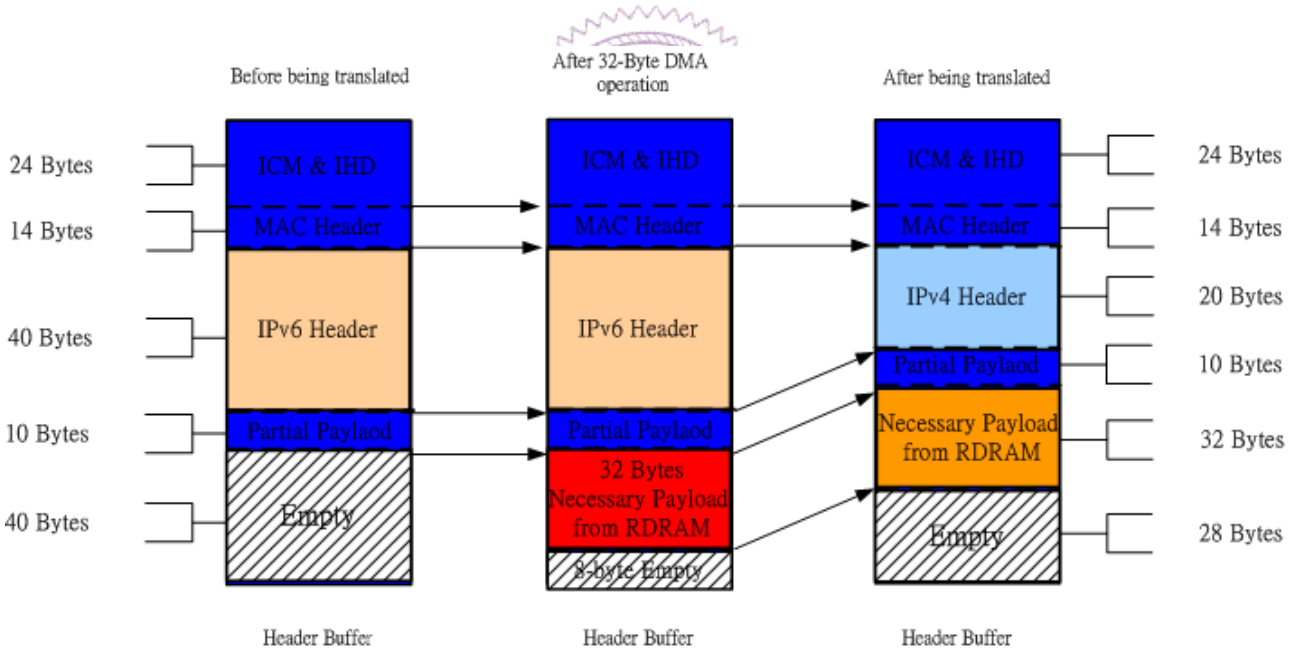


Figure 4.5 Variation of Header Buffer in Case of IPv6 to IPv4

4.5. System Initialization

During the period of initialization in our Chassis-Based NAT-PT system, the upper layer processor in control-plane setups almost the configurations for hardware components in the IQ2000 card. Besides, we accomplish one special task of constructing two mapping tables between IP address and MAC address in both IPv4 and IPv6 domains using a method we call Pre-Resolution to retrieve the MAC address of the nodes near the NAT-PT translator. Pre-Resolution sends lots of ARP Requests to the nodes in IPv4 side and NS to the nodes in IPv6 side for their possible future address resolutions. In fact, the way to handle the layer 2 packets of address resolution is not defined in a particular RFC and there are various methods for that such as Proxy Resolution in PC-Based NAT-PT and Transparent Resolution in NP-Based (Intel IXDP1200) NAT-PT [38]. These different methods of address resolution can achieve the same objective but with different costs either in implementing complexity or in system performance. Before giving the detail of Pre-Resolution in this thesis, we are now briefly introducing the Proxy Resolution and Transparent Resolution mentioned earlier in following.

- Proxy Resolution in PC-Based NAT-PT

The PC-Based NAT-PT is implemented in Linux platform. The layer 2 processing of address resolution is completed by CPU. The way it used to deal with the layer 2 packets of address resolution is to reply to the ARP Request, from one IPv4 node, who wants to get the MAC address of one IPv6 node and the NS, from IPv6 node, who wants to get the MAC address of one IPv4 node with an ARP Reply and NA carrying the MAC address of the NAT-PT translator respectively. Figure 4.6 shows the big picture of Proxy Resolution in PC-Based NAT-PT for handling the layer 2 packets.

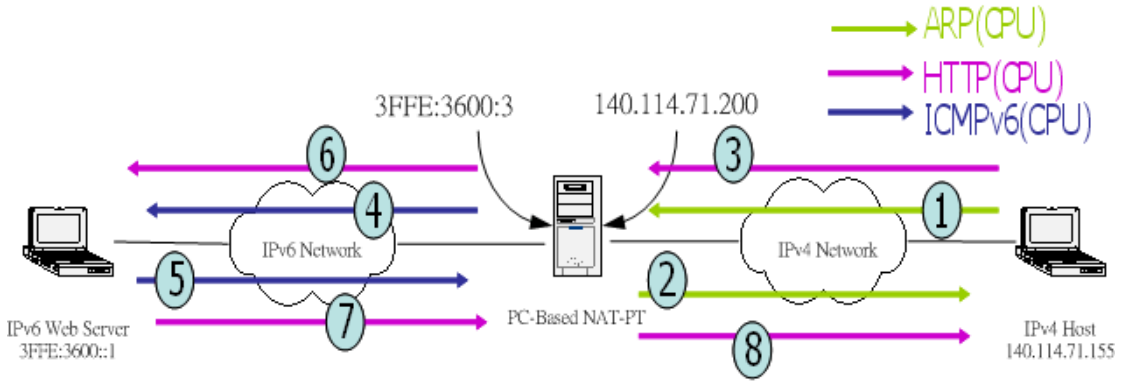


Figure 4.6 Proxy Resolution

- Transparent Resolution in NP-Based NAT-PT

The NP-Based NAT-PT we bring to compare with is implemented in Intel IXDP 1200. It puts the layer 2 processing of address resolution both in data-plane and control-plane. The way it used to cope with the layer 2 packets of address resolution is to translate the ARP Request/NS from IPv4 node/IPv6 node into corresponding equivalent NS/ARP Reply for further resolution in another address domain. In the response of either an ARP reply to IPv4 node or a NA to IPv6 node, it contains the MAC address the one it wants to communicate with. That is to say, the IPv6 node who sends the NS to NAT-PT gets the MAC address of the IPv4 node and vice versa. Figure 4.7 illustrates the operations of Transparent Resolution in NP-Based NAT-PT in IXDP 1200 for translating the layer 2 packets.

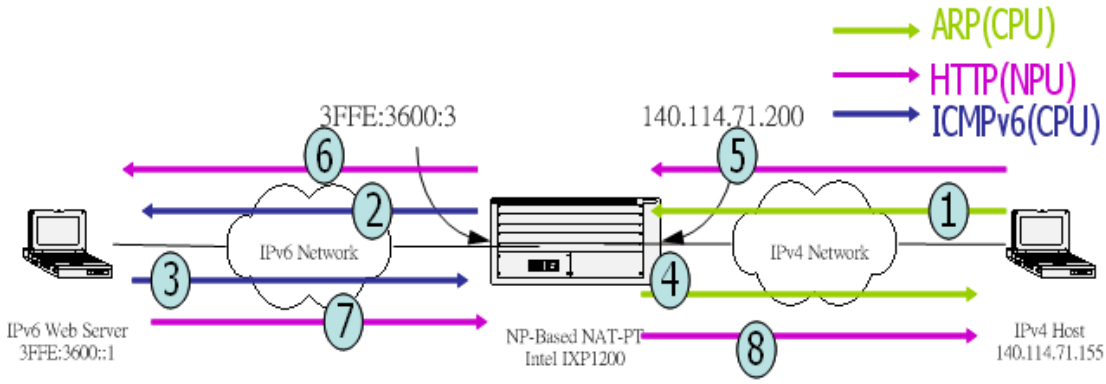


Figure 4.7 Transparent Resolution

4.5.1. Pre-Resolution in Chassis-Based NAT-PT

Since the ways of address resolution in NAT-PT implemented on either PC-Based platform or NP-Based platform must be processed by CPU completely and partially respectively, we want to accomplish the layer 2 processing all depend on Network Processor to reduce the processing time of address resolution. Therefore, we use Pre-Resolution as our policy of address resolution to cope with the layer 2 packets. Figure 4.8 describes the flow of Pre-Resolution in our Chassis-Based NAT-PT system at layer 2 processing.

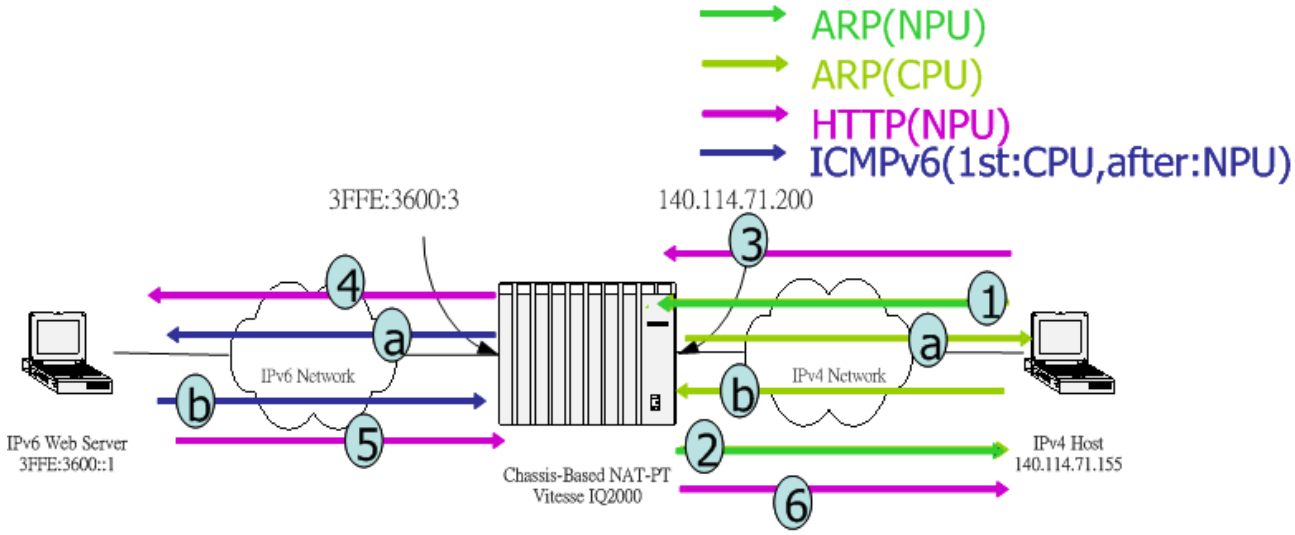
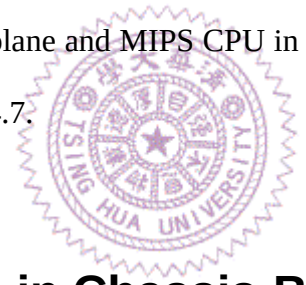


Figure 4.8 Pre-Resolution

As the Figure 4.8 illustrates, the upper layer processor sends lots of ARP Requests to IPv4 domain with the IPv4 address in the same subnet of IPv4 interface of Chassis-Based NAT-PT and NS to IPv6 domain with the IPv6 address in the same prefix of IPv6 interface of Chassis-Based NAT-PT. Therefore, the IPv4 nodes who receive the ARP Requests belong to them reply the corresponding ARP Replies including their MAC addresses to the IPv4 interface of Chassis-Based NAT-PT and the IPv6 nodes who receive the NS belong to them reply the corresponding NA including their MAC address to the IPv6 interface of Chassis-Based NAT-PT. As the FACET

receive these ARP Replies and NAs, it passes the whole packets to the upper layer processor at control-plane for putting the MAC addresses together into IPv4-MAC table and IPv6-MAC table where the MAC address in IPv4-MAC table come from those ARP Replies and the MAC address in IPv6-MAC table come from those NAs. Finally, while FACET receives the ARP Reply/NS coming from one IPv4/IPv6 node that is going to communicate with one IPv6/IPv4 node, the FACET directly replies the corresponding ARP Reply/NA including the necessary MAC address from IPv6-MAC/IPv4-MAC table to IPv4/IPv6 node. Then the IPv4/IPv6 node can directly sends the session packets to IPv6/IPv4 node through NAT-PT without indirectly address resolution in PC-Based NAT-PT and the layer 2 translation in NP-Based NAT-PT. we will give the detail comparison among these the methods of address resolutions in different platform to show out the advantages using the Pre-Resolution in next chapter. The processes of Pre-Resolution need the cooperation of FACET CPU in data-plane and MIPS CPU in control-plane. The former describes in section 4.6 while the later in section 4.7.



4.5.2. Mapping Tables in Chassis-Based NAT-PT

In our Chassis-Based NAT-PT, we have to construct three mapping tables to assist the NAT-PT system. They are one IPv4-IPv6 address mapping table and two IP-MAC address mapping tables. We use Table 4.7 to figure out the properties among these address mapping tables. Among them, IP-MAC address mapping tables are constructed during Pre-Resolution process by MIPS CPU in control-plane and made a description of how to take shape in section 4.7.

Table	Entry	Location	Function	Maintainer	Access
IPv4-IPv6	(IPv4 Address, IPv6Address)	RDRAM	Header Translation	MIPS	FACET,MIPS
IPv4-MAC	(IPv4 Address, MAC of IPv4 node)	RDRAM	Address Resolution for IPv6 node	MIPS	FACET
IPv6-MAC	(IPv6 Address, MAC of IPv6 node)	RDRAM	Address Resolution for IPv4 node	MIPS	FACET

Table 4.7 Properties of Tables in Chassis-Based NAT-PT

4.6. Processes in Data-Plane

The processes handled by FACET CPU in data-plane contain Input Descriptor modification, header translation, enqueuing process, etc. Most of them are handled by the FACET CPU, while some of them need assistance from MIPS CPU. The following subsections will detail these processes.

4.6.1. Packet Descriptor Modification

Because of the issue resulted from the difference of packet length between IPv4 and IPv6 and the size of the HEADER_SIZE described in section 4.4, we should do some modification on Packet Descriptor during the FACET processing. We have two cases to modify the Packet Descriptor depending on the direction of the packet flow in data-plane.

- IPv4→IPv6

Since we give a short value to HEADER_SIZE, we do not need to move the extra data from Header Buffer to RDRAM. So the FACET doesn't need to change the information in PLENGTH, PA, and BUFA at all. The only field we should modify is HLENGTH since the header length is 20-byte increased after header translation.

- IPv6→IPv4

While we give a policy that no matter what kind of IPv6 packet comes into IQ2000, we uniformly move 32-byte "Payload" data from RDRAM to Header Buffer. The operation results the following modification of Packet Descriptor.

```
if Packet_Type = Header                    // there is no payload in RDRAM

    New_HLEN = Ori_HLEN – 20      // header length is 20-byte decreased after
                                              header translation
```

```
        New_PLEN = 0                    // no payload in RDRAM
        PA = 0
        BUFA = 0

else if Packet_Type = Long              // there is more than 32 byte payload in
                                        RDRAM

        New_HLEN = Ori_HLEN + 12        // 32 (shift amount from RDRAM) – 20 = 12
        New_PLEN = Ori_PLEN – 32
        PA = 1                          // change the address mode of payload
        BUFA = 32                       // offset from first DBP

// Complete Type Below

else if (Ori_PLEN – 32 >= 0)            // there is payload remaining in RDRAM after
                                        32-byte data movement from RDRAM to
                                        Header Buffer

        New_HLEN = Ori_HLEN + 12
        New_PLEN = Ori_PLEN – 32
        PA = 1
        BUFA = 32

else    // there is no payload remaining in RDRAM after 32-byte data movement
           from RDRAM to Header Buffer
```

```
New_HLEN = Ori_HLEN + Ori_PLEN – 20 // Ori_PLEN is the shift amount
                                     from RDRAM
New_PLEN = 0
PA = 0
BUFA = 0
```

4.6.2. Header Translation

In our Chassis-Based NAT-PT system, we have four types of header translation: Ethernet Header, IPv4/IPv6 Header [11][12], ICMPv4/IPMPv6 Header [11][12][39][40], TCP Header, and UDP Header. In our configuration of HEADER_SIZE, all incoming packets from IPv4 side could be well processed all in the Header Buffer without extra DMA operation from RDRAM to Header Buffer for further “Payload” processing while the incoming packets from IPv6 side could not. Moreover, we let our NAT-PT system have the ability of replying ARP Request and NS messages with corresponding ARP Reply and NA message respectively in data-plane. Table 4.8 demonstrates the mappings between these messages with the view of OSI Layer.

Incoming		Outgoing		Note
IPv4 Domain	Ethernet Header	IPv6 Domain	Ethernet Header	The “Type/Length” field has to be changed
	IPv4 Header		IPv6 Header	Defined in RFC2765 (SIIT) and RFC2766 (NAT-PT)
	TCP Header		TCP Header	The pseudo header checksum needs to be re-computed
	UDP Header		UDP Header	The pseudo header checksum needs to be re-computed
	ICMPv4 Header		ICMPv6 Header	Defined in RFC2765 (SIIT), and the pseudo header checksum needs to be recomputed

IPv4 Domain	ARP Request	IPv4 Domain	ARP Reply	Defined in RFC826 (ARP)
IPv6 Domain	Neighbor Solicitation	IPv6 Domain	Neighbor Advertisement	Defined in RFC2461 (ND for IPv6)

Table 4.8 Header Translation in Chassis-Based NAT-PT System

- Ethernet Header

The *Type/Length* field in IPv4 domain is “0x0800” or “0x0806” (in case of ARP), while in IPv6 domain is “0x86DD”

- IPv4/IPv6 Header

There are several fields have to be translated between IPv4 and IPv6 header, and they are defined in the RFC2765 and RFC2766. We give a summarization of IPv4/IPv6 Header Translation in Table 4.9 and Table 4.10. Besides, Figure 4.9 and Figure 4.10 show the policies when we perform the header translation in case of receiving IPv4 packets and IPv6 packets respectively.

IPv6 Fields	value
Version	6
Traffic Class	0
Flow Label	0
Payload Length	No Fragmented:IPv4 Total Length - (IPv4 Header length + Option length) Fragmented:IPv4 Packet Total Length - (IPv4 Header Length + Option Length) + (Fragment Header Length)
Next Header	No Fragmented: the same with the protocol field in IPv4 header Fragmented: Fragment Header
Hop Limit	TTL value in IPv4 header - 1
Source Address	Low-order 32bits = source address in IPv4 header High-order 96bits = IPv6 Prefix defined in NAT-PT
Destination Address	Low-order 32bits = free IPv4 address in address pool High-order 96bits = IPv6 Prefix defined in NAT-PT

Table 4.9 IPv4 to IPv6 IP Header Translation

IPv4 Fields	value
Version	4
ToS and Precedence	0
Total Length	No Fragmented: (Payload length in IPv6 Header) + IPv4 Header Length Fragmented: (Payload length from IPv6 Header) + IPv4 Header Length
Identification	No Fragmented: Identification field in IPv6 Fragment Header Fragmented: 0
Flags	No Fragmented: 0 Fragmented: MF = MF bit from IPv6 Fragment Header & DF = 0
Fragment Offset	No Fragmented: 0 Fragmented: Fragment Offset field in IPv6 Fragment Header
Time to Live	Hop Limit in IPv6 Header - 1
Protocol	No Fragmented: Next Header field in IPv6 Header Fragmented: Next Header field in IPv6 Fragment Header
Header Checksum	Re-Calculate
Source Address	the corresponding IPv4 address in IPv4-IPv6 mapping table
Destination Address	the Lower 32bits of destination address in IPv6 Header

Table 4.10 IPv6 to IPv4 IP Header Translation

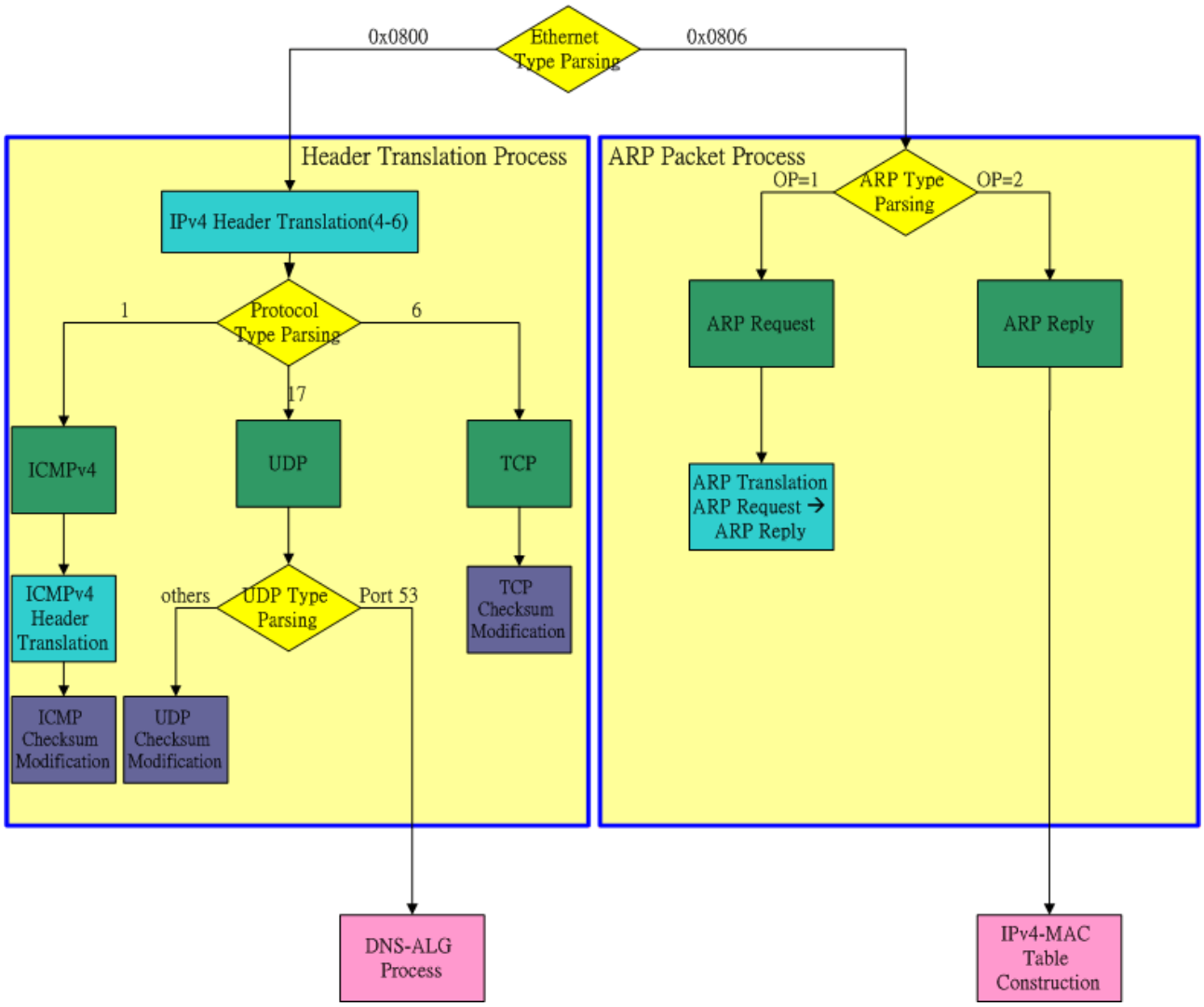


Figure 4.9 IPv4 to IPv6 Function Flow Chart

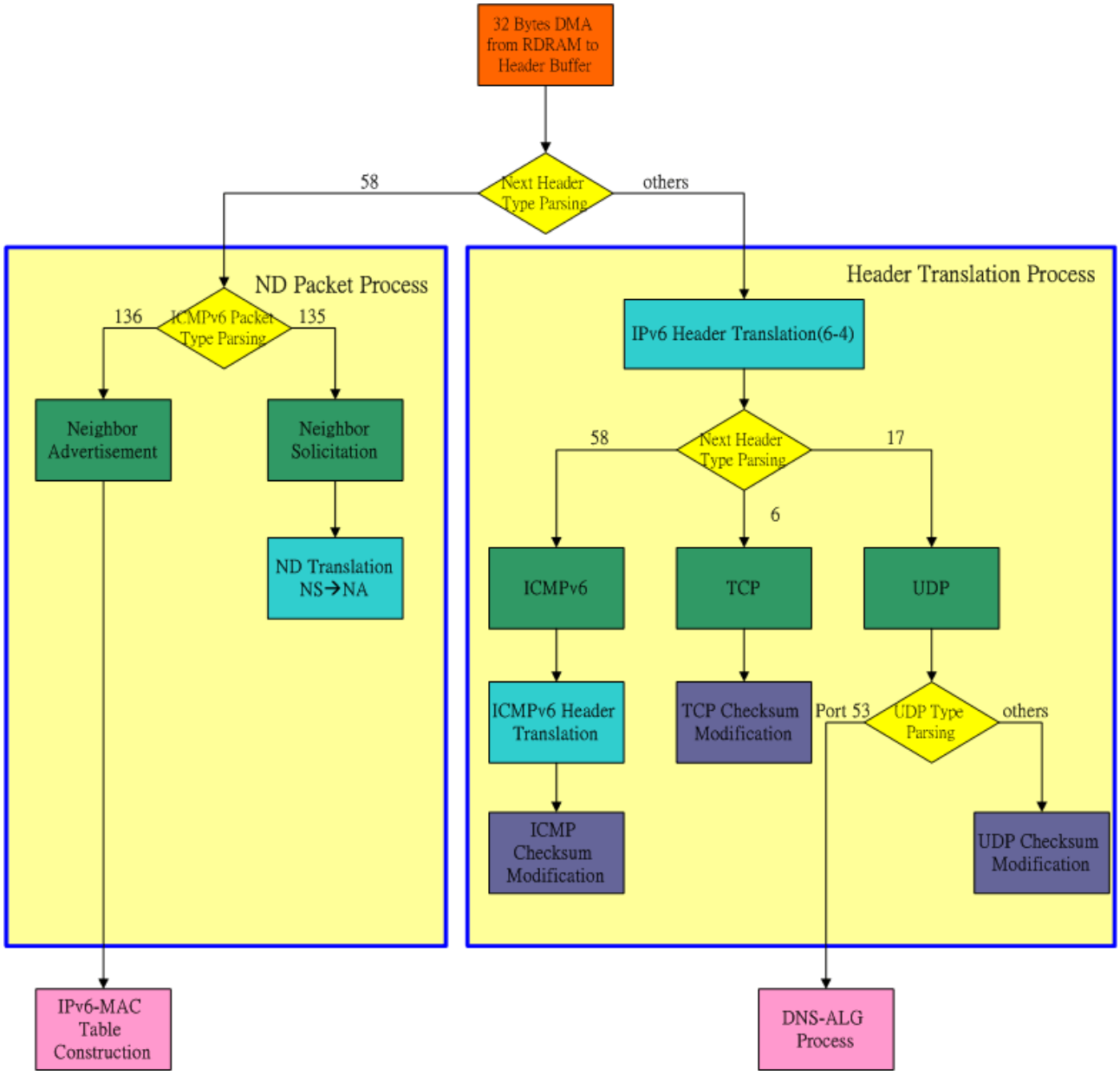


Figure 4.10 IPv6 to IPv4 Function Flow Chart

- TCP Header

Basically, we do not need to translate any field for TCP. However, the TCP pseudo-header checksum has to be re-computed because the IPv4/IPv6 address has been replaced.

- UDP Header

Similarly with TCP header, the pseudo-header checksum has to be re-computed. Moreover, if

this is a DNS service (UDP traffic with port 53), we also have to translate DNS message (A←→AAAA), and send the frame to DNS-ALG for address allocation. We have further description in next section about DNS-ALG.

- ICMPv4/ICMPv6 Header

The mapping between ICMPv4 and ICMPv6 is also defined in RFC2765. Besides the field translation, ICMPv6 also contains the pseudo-header for checksum calculation, while ICMPv4 does not.

- ARP Request/ARP Reply

Since our Chassis-Based NAT-PT system maintains one IPv6-MAC table including the address mappings in the RDRAM, we can simply translate the ARP Request coming from IPv4 domain into corresponding ARP Reply for completing address resolution. Table 4.11 shows the header translation from ARP Request to ARP Reply [36][37].

Value of ARP Request	Field	Value of ARP Reply
MAC Header		
0xFFFFFFFFFFFF	Destination Address	IPv4 node who sends the ARP Request
the MAC address of IPv4 node who sends the ARP Request	Source Address	the MAC address of IPv6 node coming from IPv6-MAC table
ARP Message		
1	Operation	2
the MAC address of IPv4 node who sends the ARP Request	Sender Ethernet Address	the MAC address of IPv6 node coming from IPv6-MAC table
the IP address of IPv4 node who sends the ARP Request	Sender IP Address	the mapped IPv4 address of IPv6 node

0x000000000000	Target Ethernet Address	the MAC address of IPv4 node who sends the ARP Request
the mapped IPv4 address of IPv6 node	Target IP Address	the IP address of IPv4 node who sends the ARP Request

Table 4.11 ARP Header Translation

- NS/NA

Similarly to ARP message translation, we also have an IPv4-MAC table implemented in memory for dealing with the address resolution from IPv6 domain. Our Chassis-Based NAT-PT system translates the NS message from a certain IPv6 node into corresponding NA message with the cooperation of FACET CPU and then sends it back. Table 4.12 shows the difference of packet field between NS and NA [35][39][40].

Value of NS	Field	Value of NA
MAC Header		
0x3333FFXXXXXX (XX: the right most 12 bit of solicited node's IPv6 Address)	Destination Address	the MAC address of who sends NS
the MAC adress of who sends NS	Source Address	the MAC address of solicited IPv4 node coming from IPv4-MAC table
IPv6 Header		
the IPv6 address of who sends NS	Source Address	the solicited node's IPv6 address
FF02::FF:XX:XX:XX(XX:the right most 12 bit of solicited node's IPv6 Address)	Destination Address	the IPv6 address of who sends NS
135	Type	136
	Checksum	Re-Calculation
	RSO	011

the solicited node's IPv6 address	Target IPv6 Address	
1	Option Code	2
1	Option Code Length	1
the MAC address of who sends the neighbor solicitation	Sender/Target Link Address	the MAC address of solicited IPv4 node coming from IPv4-MAC table

Table 4.12 ND Header Translation

4.6.3. Enqueuing Process

4.6.3.1. Enqueuing the OHD to Smart Buffer Queue

After finishing the header translation in Header Memory by FACET, we have to enqueue the OHD into the Smart Buffer Queue. In our design, we take PIM A/POM A as the input/output interface of IPv4 packet while PIM B/POM B as the input/output interface of IPv6 packet. That is to say, we should put the OHD translated from IPv4 packets into the Smart Buffer Queue B (mapping to POM B) while the OHD translated from IPv6 packets into Smart Buffer Queue A (mapping to POM A). Figure 4.11 presents the enqueue process of OHD.

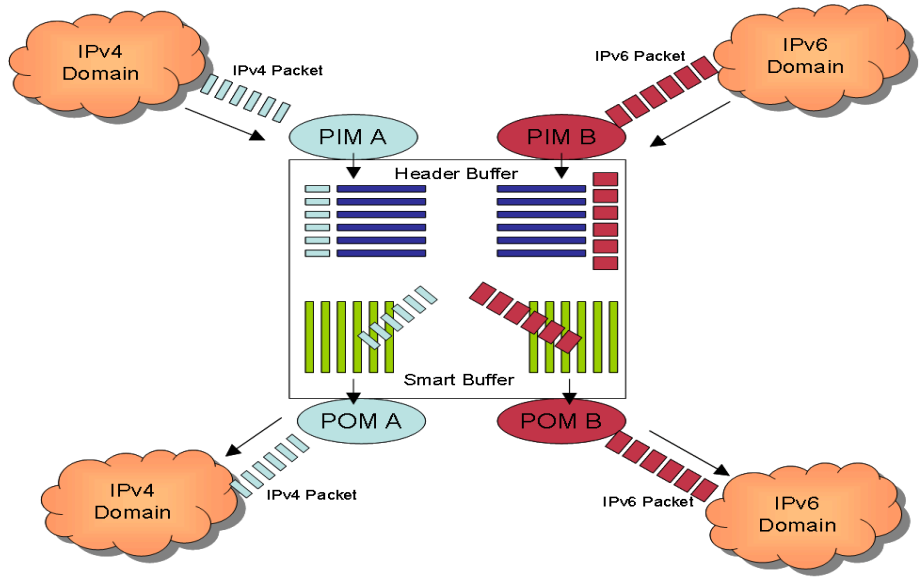


Figure 4.11 Enqueue Process

POM device detects the Smart Buffer Queue to check if OHD exists. POM retrieves the OD popped from corresponding Smart Buffer Queue to derive some packet information like packet size and memory address of “Payload” in RDRAM if has. The calculation of “Payload” address in RDRAM is different in different packet flow as Table 4.6 describes while the calculation of packet size is according its packet Type described as below.

- Header Type: No Payload

$PacketSize = HLENGTH$

- Complete and Long Type: Payload exists

$PacketSize = HLENGTH + PLENGTH$

4.6.3.2. Communication between PPE and MIPS

The Prism Driver is responsible for system configuration and being an interface between PPE in data-plane and MIPS CPU in control-plane. For communication between PPE and MIPS, the Prism Driver used four circular queues including 8 128-byte elements each in main memory as the data buffer for each of the four PPEs. The data-plane packet which needs to be processed by control-plane procedures would be like to be enqueued into one certain queue according to which PPE the data comes from. In the design of Chassis-Based NAT-PT, there are three types of packet known as ARP Reply, NA, and UDP with port 53 must be passed to the corresponding communication queues for MIPS processing. We define three groups of four circular queues to one type of packet each. Figure 4.12 shows the process of communication between PPE and MIPS of these three types of packets.

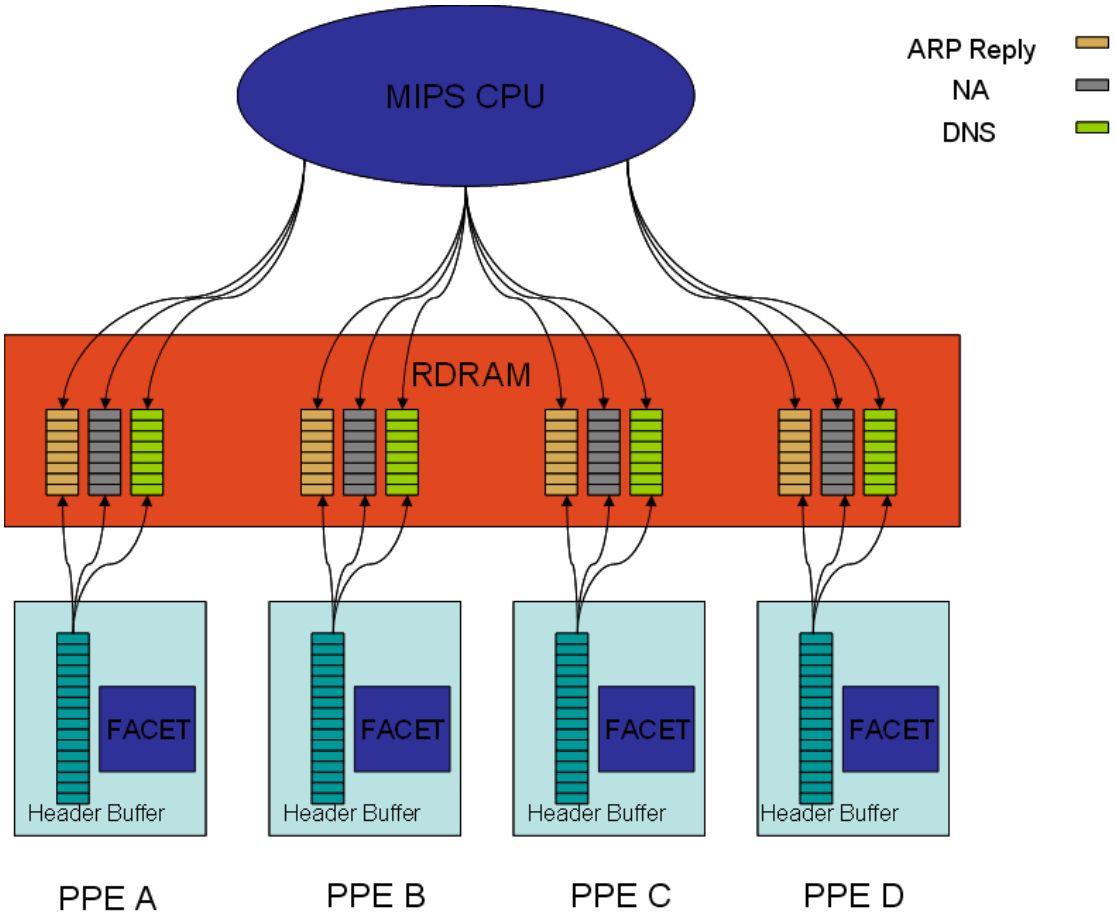


Figure 4.12 Communicate Between PPE and MIPS

4.7. Processes in Control-Plane

4.7.1. IP-MAC Table Construction

As described in section 4.5.1, there are two IP-MAC address mapping tables constructed in main memory since the Chassis-Based NAT-PT takes Pre-Resolution as the policy for processing the layer 2 address resolution either from IPv4 domain or IPv6 domain. Following we are detail explaining how the two tables become and what the two tables contain.

- IPv4-MAC address mapping Table

  During the period of system initialization, MIPS CPU sends lots of ARP Request message to

the network the Chassis-Based attaches to with the packet format like Table 4.10 presents. Each node in the network replies the ARP Reply with their MAC address to the IPv4 interface of Chassis-Based NAT-PT. The PPE puts these ARP Reply messages into the communication queue waiting for MIPS processing. MIPS will polls its queue to check if something exists. The MIPS CPU extracts the MAC addresses form those ARP Reply messages and adds them into the proper entry in IPv4-MAC table.

- IPv6-MAC address mapping Table

Similarly to the IPv4-MAC table, MIPS CPU sends lots of NS messages to network the NAT-PT attaches to with another packet format like Table 4.11 presents. After the similar operation of enqueuing the NA packets from IPv6 domain to the queue in main memory, MIPS extracts the MAC address from “Target Link Address” field in NA and adds them into the IPv6-MAC table.

When the process of table construction finished, the Chassis-Based NAT-PT system may well processing the layer 2 resolution from both IPv4 and IPv6 domains by accessing the IPv6-MAC and IPv4-MAC tables respectively.

4.7.2. DNS-ALG

There is no huge difference from PC-Based NAT-PT, NP-Based NAT-PT (Intel IXDP1200) and Chassis-Based NAT-PT (Vitesse IQ2000) for implementing DNS-ALG, since it were all written in C language but only on different platform. For DNS-ALG in our Chassis-Based NAT-PT, we have to rebuild OHD since the packet size of translated DNS Reply message will be changed. Figure 4.13 presents the processing flow of DNS-ALG in control-plane.

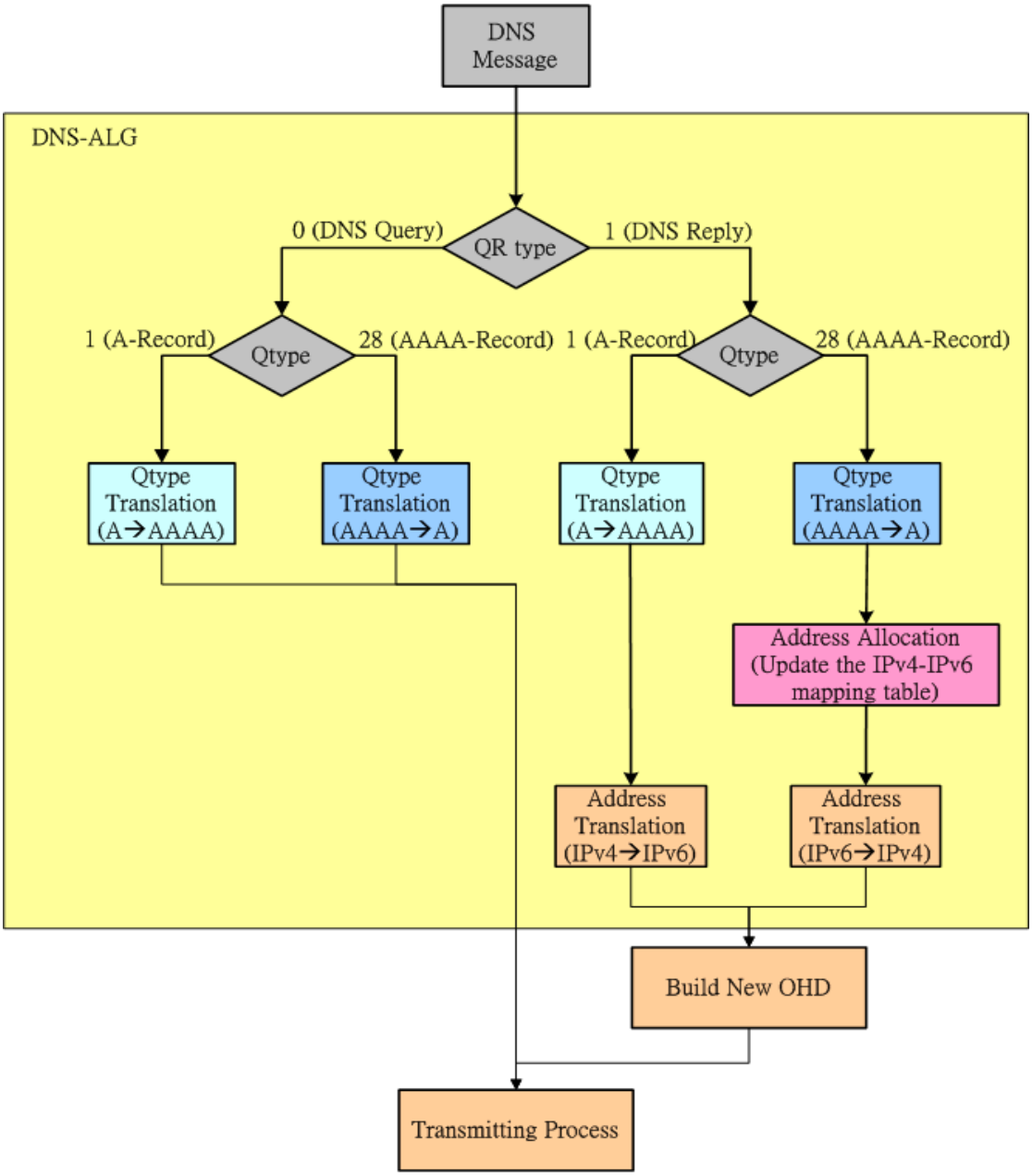


Figure 4.13 DNS-ALG Processing Flow Chart