Chapter 2.

Related Work

This chapter is separated into two sections. We will briefly discuss the currently available transition mechanisms, including Tunneling, Dual Stack, and NAT-PT in the first section. The second section introduces the Vitesse IQ2000 Network Processor, which is the platform we used to implement in our thesis.

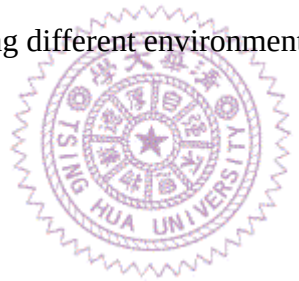
2.1. Transition Mechanism

Since the migration from IPv4 to IPv6 will not happen overnight, we expect to go through a transition period that might be last several years during which IPv4 and IPv6 network will coexist, and communication should be possible to across the boundary of the coexisting network. V6ops, IPv6 Operation working group of IETF, which took the responsibility of the former working group NGTrans (Next Generation Transition), has been working to resolve the problems that may arise during the transition period. Therefore, a strong, flexible set of IPv4-to-IPv6 transition and coexistence mechanisms are needed at the duration. A wide range of mechanisms are proposed and implemented, basically falling into three categories:

- Dual Stack mechanisms, which allow IPv4 and IPv6 to coexist and talk to each other in the same mixed networks.
- Tunneling mechanisms, which make two IPv6 sites or islands communicate with each other across IPv4 infrastructure by encapsulating the IPv6 packets inside the IPv4 packets.
- Translation mechanisms, which use IPv4-IPv6 protocol translation to ensure IPv4-only hosts to

communicate with IPv6-only hosts transparently and vice versa.

However, there is no perfect mechanism able to resolve all of the issues coming from the migration and coexistence between IPv4 and IPv6, these mechanisms may have its own advantages and disadvantages for different network environment. Tunneling will permit making a virtual link between IPv4 and IPv6 networks and vice versa. For implementing dual stack, a node should have both IPv4 and IPv6 protocol stack. To build a directly connection between IPv4-only hosts and IPv6-only hosts, we must adopt the dual stack or translation mechanisms. In our implementation, we take NAT-PT as the main translation technique since it can make IPv4-only hosts and IPv6-only hosts directly communicate with each other without any configuration and modification to themselves. Before detail describing NAT-PT, let us take a brief introduction to the various methods of Tunneling and Dual Stack for suiting different environment.



2.1.1. Dual Stack

A host with both IPv4 and IPv6 protocol stacks calls a dual stack host which is the key implementation during the early deployment of IPv6. A dual stack host can make communication with IPv4-only host through IPv4 protocol stack and IPv6-only host through IPv6 protocol stack. Figure 2.1 shows the basic operation of a dual stack host for communicating with the other devices. Which protocol will be used to make connection depends on the type of DNS response since the dual stack host will send DNS queries of both A and AAAA type records to DNS server for domain name lookup. When both are available, the dual stack host will usually choose the IPv6 path, which increases the value and power of IPv6 network by creating more users.

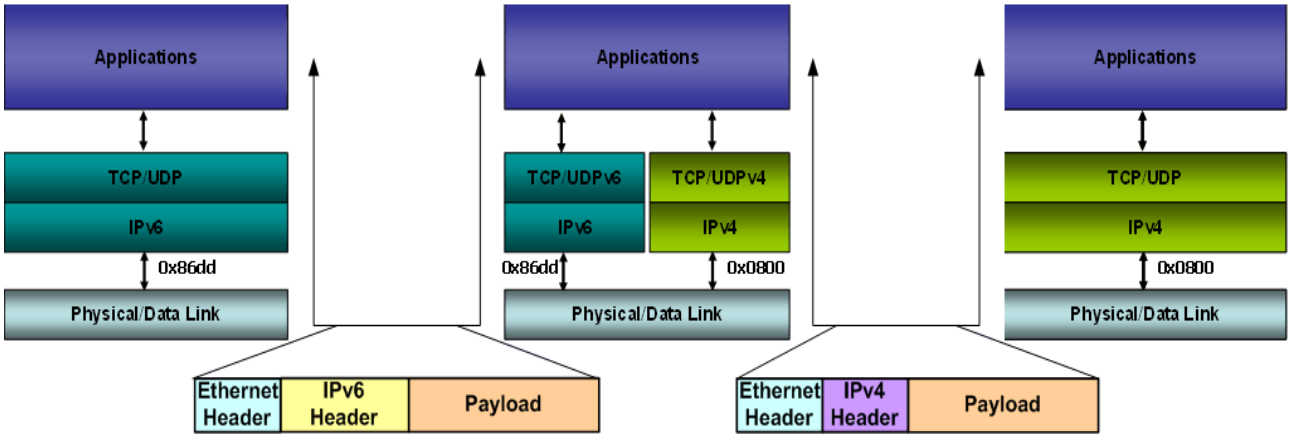


Figure 2.1 Dual Stack Mechanism

2.1.2. Tunneling

The major purpose of tunneling is to connect two separated IPv4 islands by an IPv4 infrastructure. Its key idea is that carrying the IPv6 packets inside the IPv4 packets to make two isolated IPv6 hosts communicate with each other through an undesirable IPv4 network. When sending an IPv6 packet, it must be encapsulated as payload with a protocol number 41 in the additional IPv4 header. When receiving a packet with a protocol number 41 at the recipient, it will de-capsulate the extra IPv4 header and proceed to process the remaining IPv6 packet as normal. A tunnel works as a one-hop link layer operation.

There are many tunneling protocols proposed to resolve this issue of connecting IPv6 networks over an IPv4 routing infrastructure and falling into two types: Configured Tunneling and Automatic Tunneling, described in RFC 2893.

- Configured Tunneling

  A configured tunnel needs exceptional configuration of the pair of tunnel endpoints residing in the edge of IPv4 network. We can not derive the IPv4 address of the tunnel endpoint from the encoded IPv6 source or destination address. Figure 2.2 shows the big picture of configured

tunneling.

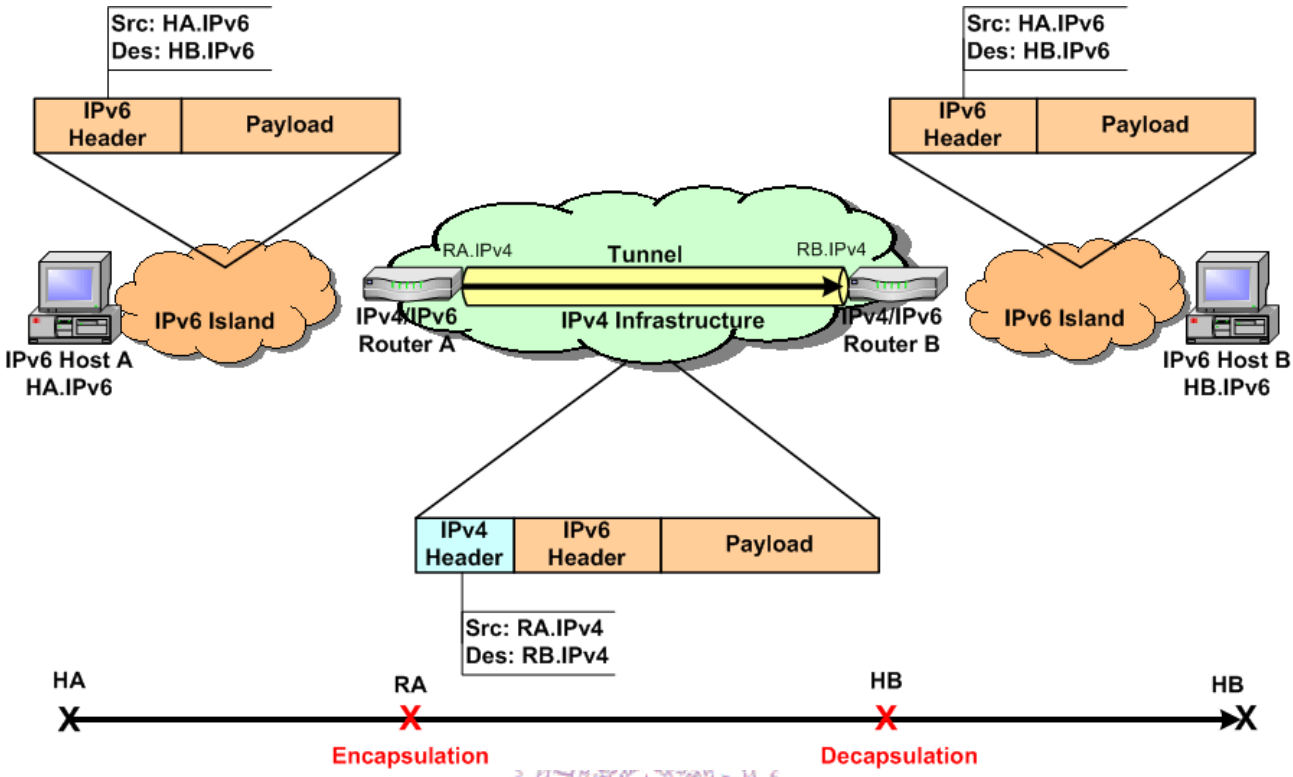


Figure 2.2 Configured Tunneling

- Automatic Tunneling

  In opposition to configured tunneling mentioned above, automatic tunneling encapsulated the egress IPv6 packets with an IPv4 compatible IPv6 address as its destination address in the additional IPv4 header automatically without any configuration of the pair of border routers. Among which, IPv4 compatible IPv6 address indicates the all-zero prefix with a globally IPv4 address in the low-order 32-bit. Figure 2.3 illustrates the basic concept of the automatic tunneling.

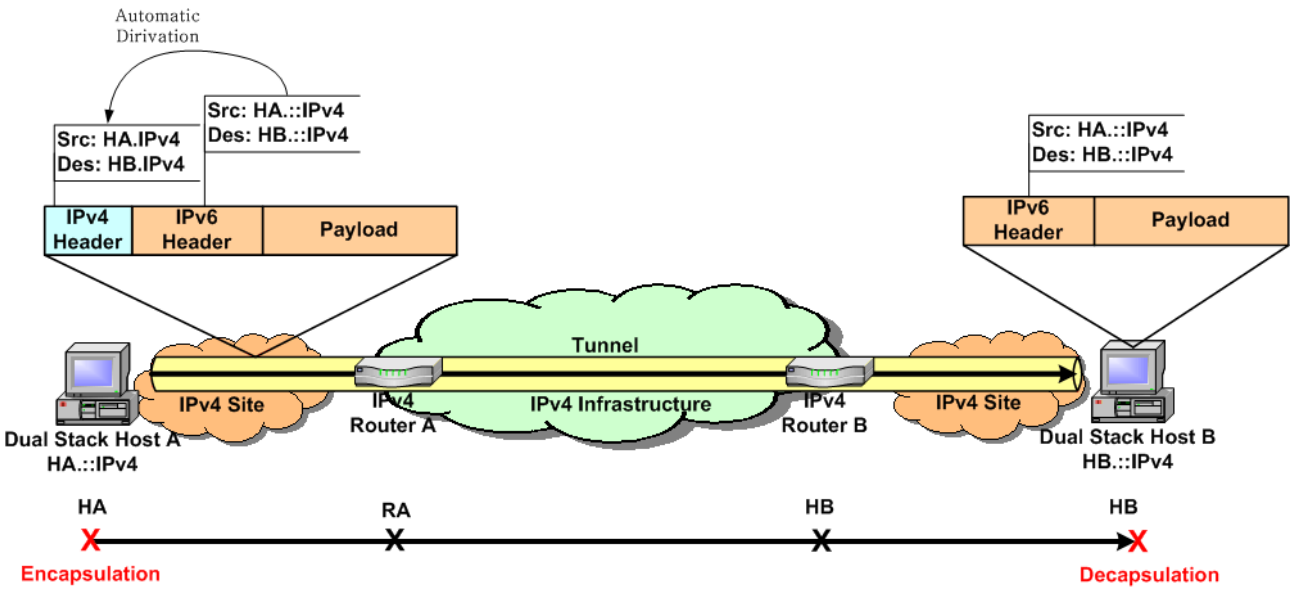


Figure 2.3 Automatic Tunneling

Automatic tunneling usually works with the configured tunneling mechanism for sending IPv6 packets with extra IPv4 header to both IPv4 compatible network and native IPv6 network across the IPv4 routing infrastructure. Except the two major tunneling methods, there are many enhanced tunneling mechanisms deriving from the configured tunneling and automatic tunneling. We will describe some of them as follows.

2.1.2.1. 6to4 Tunnel

6to4 tunneling mechanism is described in RFC 3056 which defines the following terms:

- 6to4 host

  An IPv6 enabled host which is configured at least one 6to4 address derived from the 6to4 prefix (2002::/16) and a routable IPv4 address of the 6to4 router attached to IPv4 network.

- 6to4 router

A dual stack router which is assigned an routable IPv4 address and automatically advertising a special prefix of 2002:V4ADDR/48 to the 6to4 hosts at the same link.

- 6to4 relay router

  A dual stack router which works similarly to 6to4 router except forwarding the traffic between 6to4 router and native IPv6 network.

The 6to4 mechanism tunnels the IPv6 traffic from 6to4 networks to others or native IPv6 network relying on 6to4 routers and 6to4 relay routers respectively. Each 6to4 network has a special prefix including the IPv4 address of its 6to4 router. Therefore, we can easily obtain the address of tunnel endpoint at the border of IPv4 infrastructure.

2.1.2.2. 6over4 Tunnel

The 6over4 mechanism which is described in RFC 2529 makes communications between dual stack isolated hosts across an IPv4 infrastructure which must be IPv4 multicast-enabled. It defines a set of mappings between IPv4 multicast addresses and IPv6 multicast addresses to do neighbor discovery process (Neighbor Solicitation, Neighbor Advertisement, Router Solicitation , Router Advertisement) over IPv4 network as over IPv6 network for resolution and auto-configuration. Therefore, a 6over4 enabled host which has a IPv6 link-local address embedded an IPv4 address in the lower 32-bit with a prefix of FE80::/64 can communicate with the other hosts which are also 6over4 enabled or in remote IPv6 network separated by an IPv4 multicast-enabled network environment. It means that the usage of multicast makes the 6over4 hosts and the 6over4 router, applying the same mechanism as a 6over4 host does, attached to both kinds of networks live in the same physical link so that they have fully IPv6 functionality.

2.1.2.3. Tunnel Broker

Tunnel broker as described in RFC 3053 is a mechanism designed for users who want to join the IPv6 network but are isolated from other native IPv6 networks. There are three main components in the tunnel broker system namely tunnel client, tunnel broker, and tunnel server. At the beginning of setting up the tunnel, a tunnel client which should be dual stack sends the tunnel authentication and request based on HTTP protocol to the tunnel broker which supports the web-based commands for initiate the connection to remote IPv6 network. The configurations will be set on both tunnel server and DNS server by tunnel broker according to the information coming from tunnel client after receiving the requests from tunnel client. The tunnel broker is a manager which is responsible for user registration and tunnel activation in the tunnel broker system. We state the detail steps about creating a tunnel below.

1. The tunnel client sends a request to tunnel broker for authentication and the tunnel broker authorizes the request with an AAA authorization.
2. The client delivers its request with some important information like its IPv4 address, function, and the IPv6 prefix length needed to tunnel broker.
3. The tunnel broker picks up an IPv6 prefix with proper length to be allocated to the requesting client and one of the tunnel servers to activating the tunnel service with proper configurations.
4. The tunnel broker registers the IPv6 addresses of requesting clients in the DNS with corresponding domain names.
5. The tunnel server sends the configuration information to the selected tunnel server to enable the tunnel service.
6. The tunnel is established between the tunnel client and tunnel server.

2.1.3. NAT-PT with ALGs

2.1.3.1. Introduction to NAT-PT

NAT-PT builds on the concept of the IPv4 NAT technique for address translation plus protocol translation using SIIT (Stateless IP/ICMP Translation) to provide the IPv4-IPv6 end-to-end communication. NAT-PT is a transparent gateway that translates the IPv4/IPv6 packets into semantically equivalent IPv6/IPv4 packets. Figure 2.4 states the simple operation of NAT-PT. Since the NAT-PT maintains the state information for each session, the session packets have to be routed through the border NAT-PT device. In terms of NAT, NAT-PT device makes bidirectional connection and application services possible between both different networks with an extra ALG (application Level Gateway) to deal with the payload data embedding IP address information. There are several main topics in NAT-PT design illustrated as following sections.

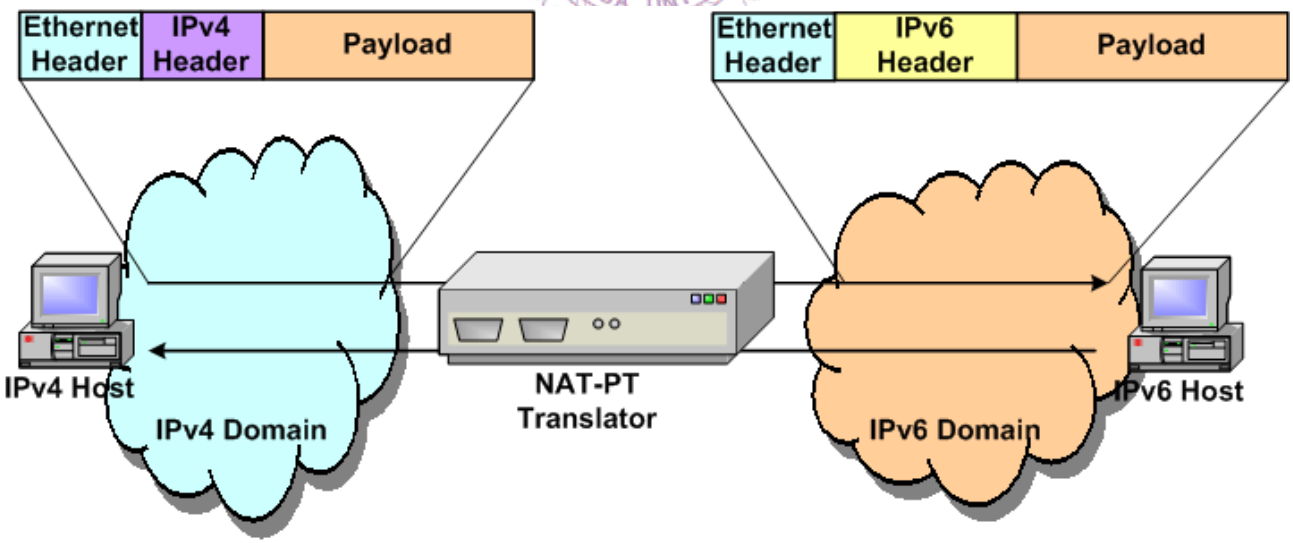


Figure 2.4 NAT-PT Translator

2.1.3.2. Network Address Translation

The first main operation of NAT-PT is network address translation which is similar to IPv4 NAT operation. NAT-PT transparently translates the packets by the schemes of providing the temporary IPv4 address for outgoing IPv6 packets to bind the internal IPv6 address with the external IPv4 address and providing the temporary ::/96 prefix for incoming IPv4 packets to bind the external IPv4 address with internal IPv6 address. While connections are originated, NAT-PT uses the globally unique IPv4 address pool for dynamic assignment to each IPv6 hosts and constructs an IPv4-IPv6 address mapping table for static or dynamic address translation.

2.1.3.3. Protocol Translation

Protocol translation built on the top of SIIT algorithm, which specifies a bidirectional translation between IPv4 and IPv6 packets including IP header translation (IPv4 Header ←→ IPv6 Header) and ICMP header (ICMPv4 Message ←→ ICMPv6 Message) translation and ignoring IPv6 extension header and IPv4 option header, is the second part of NAT-PT operation.

2.1.3.4. Application Level Gateway

NAT-PT is enough to supply the transparent communication between IPv4 and IPv6 hosts for most of the popular Internet services like WWW, E-mail, and Telnet. For some special Internet services such as DNS and FTP which embeds the IPv4/IPv6 address information in the payload part, we needs relative application level gateway like DNS-ALG [26][27][28] and FTP-ALG [29][30] to give assistance for performing corresponding payload translation respectively. We are leaving the detail of DNS-ALG and FTP-ALG in Chapter 3.

2.1.3.5. Why NAT-PT

We have a conclusion that there is no mechanism being able to support overall conditions during deploying the IPv6 environment. The transition mechanisms that we have learned are always used for a dedicated situation to the communication between IPv4 nodes and IPv6 nodes. The tunneling technique described above is usually adopted to enable the connectivity of two isolated IPv6 domains. However, in early deployment of IPv6, the main applications and services are absolutely provided by the IPv4 network so that the IPv6 users still need to wildly access the traditional IPv4 based network and services. The dual stack approach can very useful for supporting both kinds of applications and services during the period for IPv6 being integrated with IPv4. Thus, we have a forecast that a main disadvantage of this method will soon emerge from its rapid depletion of IPv4 addresses. The problem is not completely solved while the public address allocation of IPv4 is needed for a new machine in the network when this solution is used alone. Therefore, we need a robust strong mechanism to ensure the bidirectional communication between the IPv4-only and IPv6-only domains for the continued access to traditional IPv4 network and services plus native IPv6 access to IPv6 services. NAT-PT described in RFC 2766 has appeared to meet this demand. NAT-PT can be implemented on a router platform or PC-based platform like Linux/FreeBSD/Windows. In this thesis, we implement the NAT-PT on a network processor-based platform.

2.2. Vitesse IQ2000

The Vitesse IQ2000 [31] is an intelligent network processor platform which we are used to implement NAT-PT system. It consists of four packet processing engines for data-path packet processing and one MIPS host interface connected to off-chip MIPS CPU processing control-path packet. IQ2000 has two main interfaces named FOCUS Interface and MIPS Host Interface for data transmission, receiving from PIM (Packet Input Module) and transferring from POM (Packet

Output Module) at high-speed bandwidth, and connecting a MIPS CPU to process slow-path packets. Additionally, IQ2000 maintains a powerful packet management system including BAD (Buffer Allocation/De-allocation Module), OM (Order Manager), and SB (Smart Buffer) for data buffer optimization and efficient memory utilization. Before illustrating the details of our system design and implementation, we are going to a brief introduction of the Vitesse IQ2000 network processor. With the respect to functionality, the key components of IQ2000 can be classified into three categories: data flow module, packet processing module, and system module, those will be depicted in turn in the following subsections. Figure 2.5 shows the block diagram of the IQ2000.

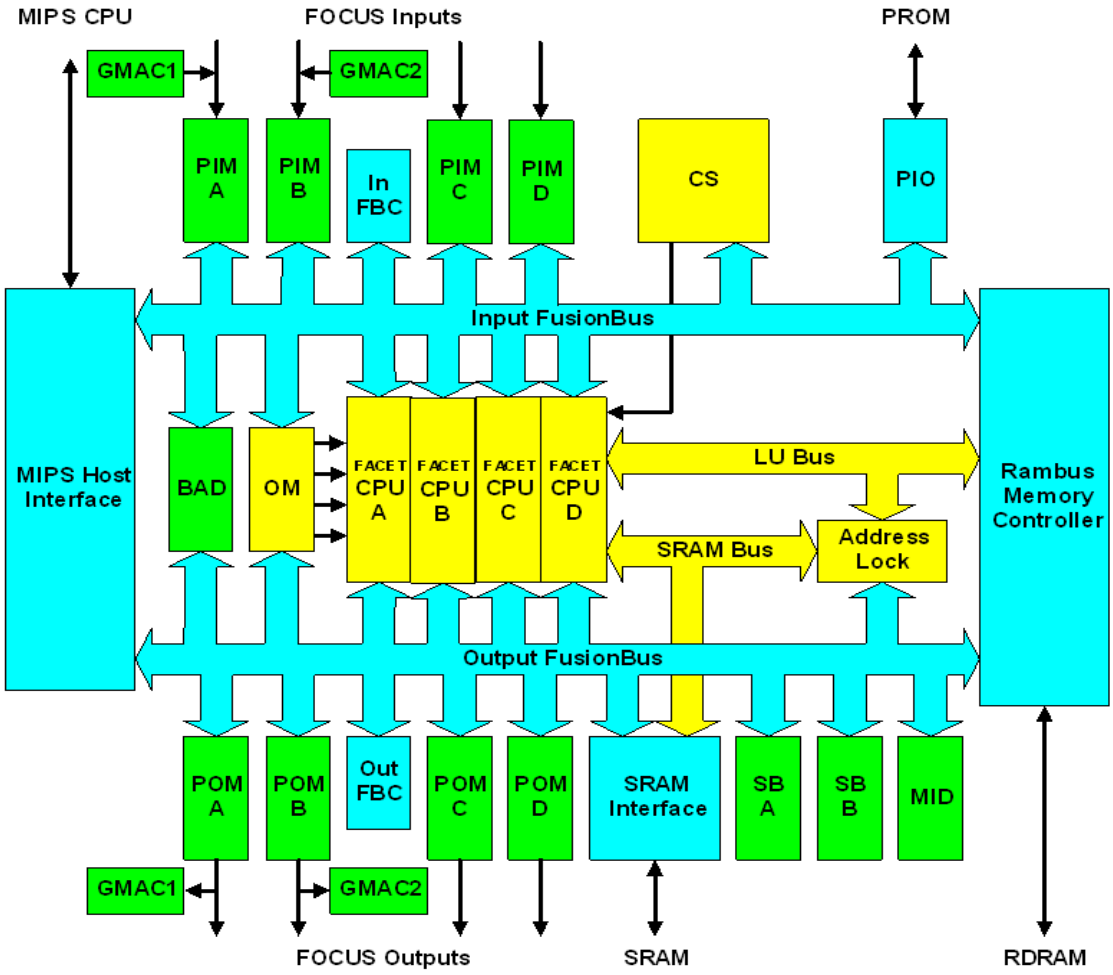


Figure 2.5 Block Diagram of IQ2000

2.2.1 Data Flow Modules

2.2.1.1. FOCUS Interface

The FOCUS Interface is a point-to-point interface used to transfer network data efficiently between the IQ2000 network processor chip and peripheral interface chip and switch fabric using in the chassis-based platform on which NAT-PT implements for connecting multiple IQ2000 modules. For transferring over the FOCUS Interface, the packets are divided into data elements called FOCUS Cells (FCELLs) sized of 1 to 128 bytes long while receiving and are reconstructed to the original complete packets while transmitting.

2.2.1.2. Packet Input Module (PIM)

The IQ2000 contains four packet input modules (PIM) named PIM A, PIM B, PIM C, and PIM D where PIM A and PIM B function as Giga port for connecting to each FOUCUS Interface. Each PIM acts as a bridge from FOCUS Interface to other IQ2000 processing modules for managing the data transfer over the Fusion Bus. The main operation of PIM is to transfer the Input Header Data (IHD), derived from the process of attaching an extra Input Descriptor (ID) to the first FCell, to packet process engine (PPE) across the Order Manager (OM) for processing and optionally deliver the other FCells, belonged to the same packet with the first FCell, to the main memory according to the FCell type indicated by the FCell header attached in the first FCell.

2.2.1.3. Packet Output Module (POM)

Like PIM mentioned earlier, the IQ2000 contains four identical packet output modules (POM) named POM A, POM B, POM C, and POM D for connecting to each FOCUS Interface. These modules play the role of outlet for the IQ2000 to manage the data transfer from internal IQ2000

components to devices on FOCUS Bus. The POM maintains up to 66 queues including 64 normal queues, one AUX Channel queue and one discarded queue for packets transferring or dropping. The major function of POM is initiating a new transfer of a single packet by selecting one FCELL, the first segment of the packet, which is retrieved from Smart Buffer (SB) among those various queues. It is notified by detecting the SB with a reply of the output descriptor, removed from the output header data, which contains the packet information including the packet size and the actual address of the packet in main memory. The process of the complete packet transfer will be done by using the information included in output descriptor for payload retrieving and packet reconstruction.

2.2.1.4. Buffer Allocation/De-allocation (BAD)

The IQ2000 uses a special module named Buffer Allocation/De-allocation (BAD) to manage the memory usage. We take the view of the available memory for legal access to IQ2000 components as eight pools, named Data Buffer Space (BSPC) each, which can be access by a special data structure called Data Buffer Pointer (DBP) maintained by BAD. The IQ2000 components use the BSPC base address and DBP to access a specific location in memory and Figure 2.6 shows the buffer allocation in main memory. The BSPC and DBP specify the particular data buffer space and particular data buffer in main memory. The actual address of the location in memory is determined as:

Address = { BSPC[31:20], 00000h } + { DBP[15:0],00h}

.

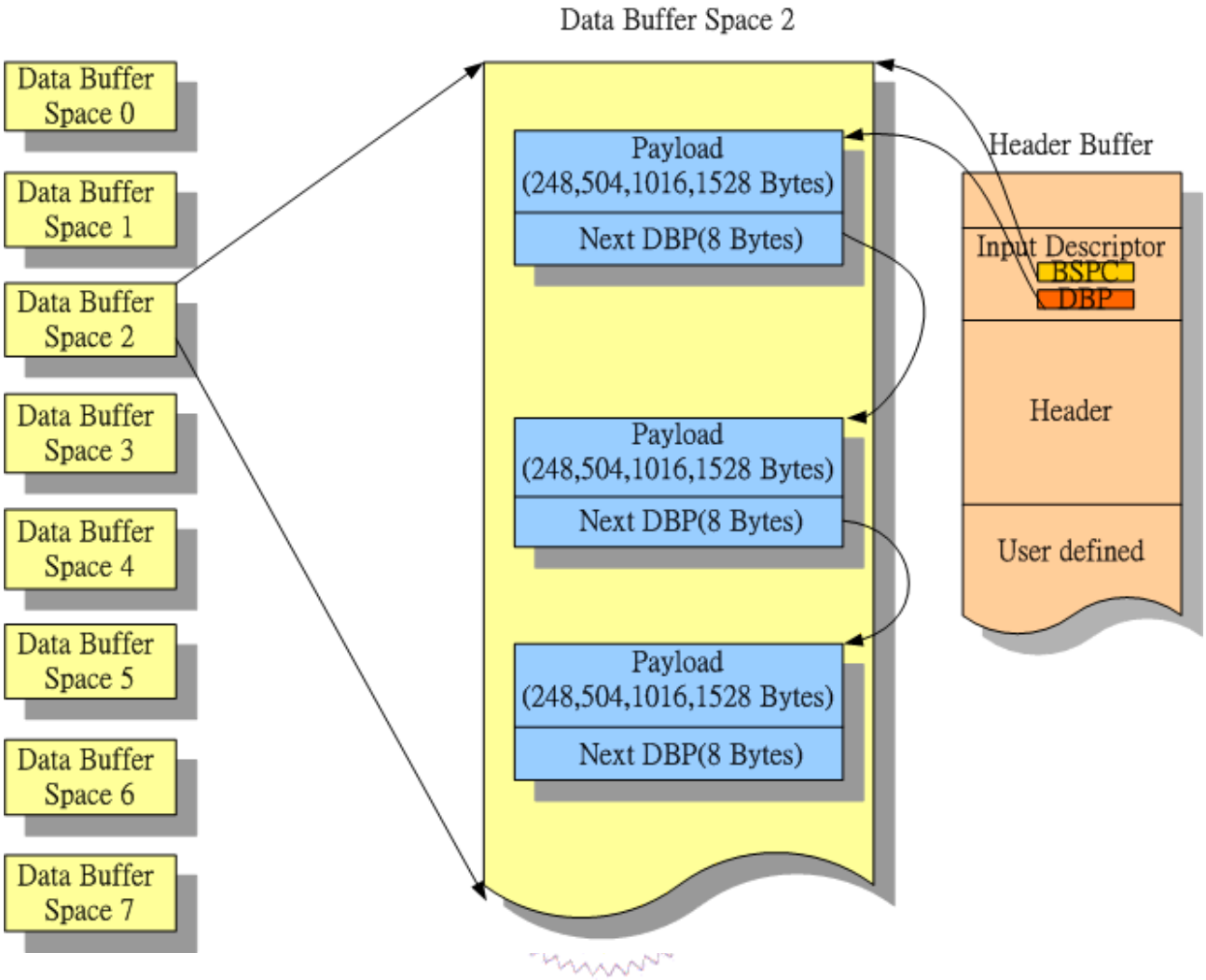


Figure 2.6 Buffer Allocation in main memory

2.2.1.5. Smart Buffer Module (SB)

The IQ2000 supplies a module called Smart Buffer (SB) for enqueue process which the packet processing engine or MIPS CPU put the Output Header Data (OHD) into the Output Header queue after packet processing and waiting for output transmission by the POM. The SB saves the output headers in either their local memory or in main memory while SB uses pointers to access them. In IQ2000, each Smart Buffer has a mapping to POM device, and that is SB A ←→ POM A, SB B ←→ POM B, SB C ←→ POM C and SBD ←→ POM D.

2.2.2 Packet Processing Modules

There are four packet processing engines (PPEs) including 200 MHz RISC CPU known as FACET, Local Memory such as Data and Header Memory, special purpose coprocessor like Lookup and DMA coprocessor. In the following subsections, we will just describe the three key components related to our system implementation about Micro-code [34] coding.

2.2.2.1. FACET CPU

There are four embedded RISC FACET CPU in the IQ2000 system used to process fast-path packets. Each of them includes independent 32 registers for their 5 contexts, 4 for user contexts to deal with general packet processing and 1 for kernel context to handle exceptions and interrupts, to active simultaneously. The FACET CPU is not only a stander-based RISC CPU but also minuses a special purpose instruction set for network data processing. The programming language for FACET CPU is called micro-code, which is assembly like and need additional knowledge of IQ2000 hardware architecture. This makes the IQ2000 system to be a programmable device for developer to implement various applications according to their needs. Figure 2.7 gives the basic block diagram of FACET CPU.

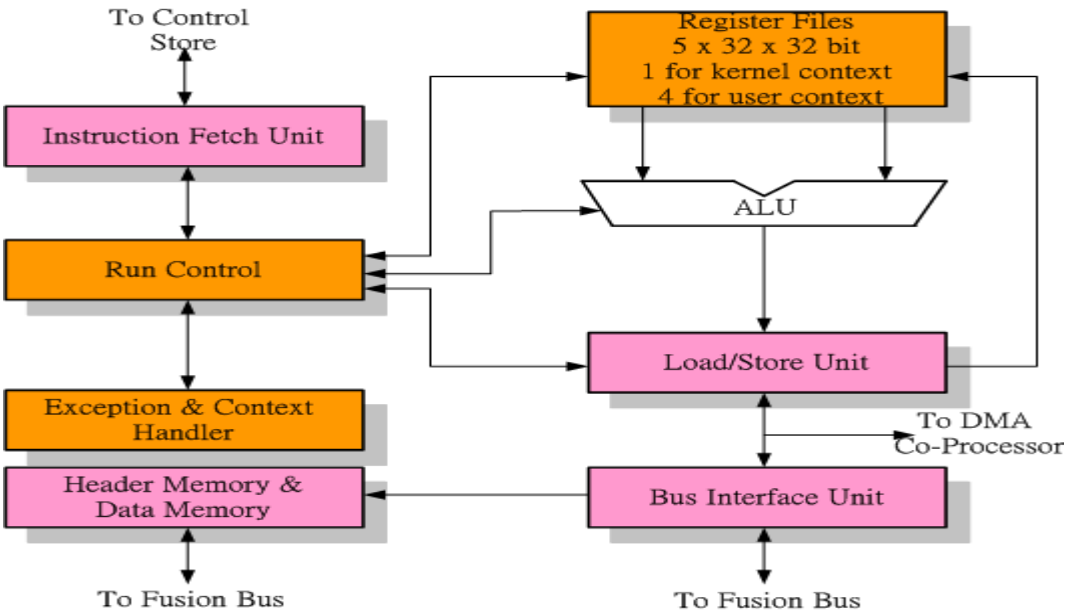


Figure 2.7 The Block Diagram of FACET CPU

The FACET is also a five-stage pipelined processor with independent buses for instruction, data and coprocessor access. It has a micro-buffer which can hold up to 16 instructions with intelligent instruction lookahead and branch prediction. The programmers can use these benefits to enhance the system performance.

2.2.2.2. Data and Header Memory

Each of the four FACET processors has its own 4KB local memory divided equally into Data memory and Header memory. The Data memory is generally used for global data storage by FACET or MIPS CPU and the Header memory is typically allocated as 16 header buffers of 128 bytes each for input header processing from PIM or MIPS CPU.

2.2.2.3. DMA Coprocessor

The DMA coprocessor supports block transfer of header and data buffers for FACET CPU with various values of DMA units between local memory and some system components like SB or main memory. On the other hand, the DMA transfers may occur in parallel with CPU operation so that can speed up the system performance.

2.2.3 System Modules

2.2.3.1. MIPS Host Interface

IQ2000 uses a interface called MIPS Host Interface for the communications between the external MIPS CPU used to control plan packet processing and the internal IQ2000 components. It supports external processor with high-speed access to FusionBus for system configuration, interrupt handling and high-layer packet processing in traditional programming way.

2.2.3.2. Fusion Bus

The FusionBus provides a general connection between modules in IQ2000. There are two kinds of FusionBus named Input FusionBus and Out FusionBus both transferring data from MIPS Host Interface and main memory but receiving packets from each PIM and transmitting packets to each POM respectively.