2.1 Router Architectures: Basic Forwarding Functions

An overview of routers and how they are architected for the purpose of packet forwarding and handling routing protocols. Traditionally, routers have been implemented purely with software running on a general-purpose personal computer (PC) with a number of interfaces. Such a device can receive packets on one of its interfaces, perform routing functions, and send packets out on another of its interfaces. As the Internet grew over the years, the type and size of routers changed, since routers based on general purpose PC architectures are limited by the performance of the central processor and memory. Fortunately, advances in silicon technology made it possible to build hardware-based routers capable of handling high incoming data rates.

FUNCTIONS OF A ROUTER

A router must perform two fundamental tasks: *routing* and *packet forwarding*, as shown in Figure 2.1.

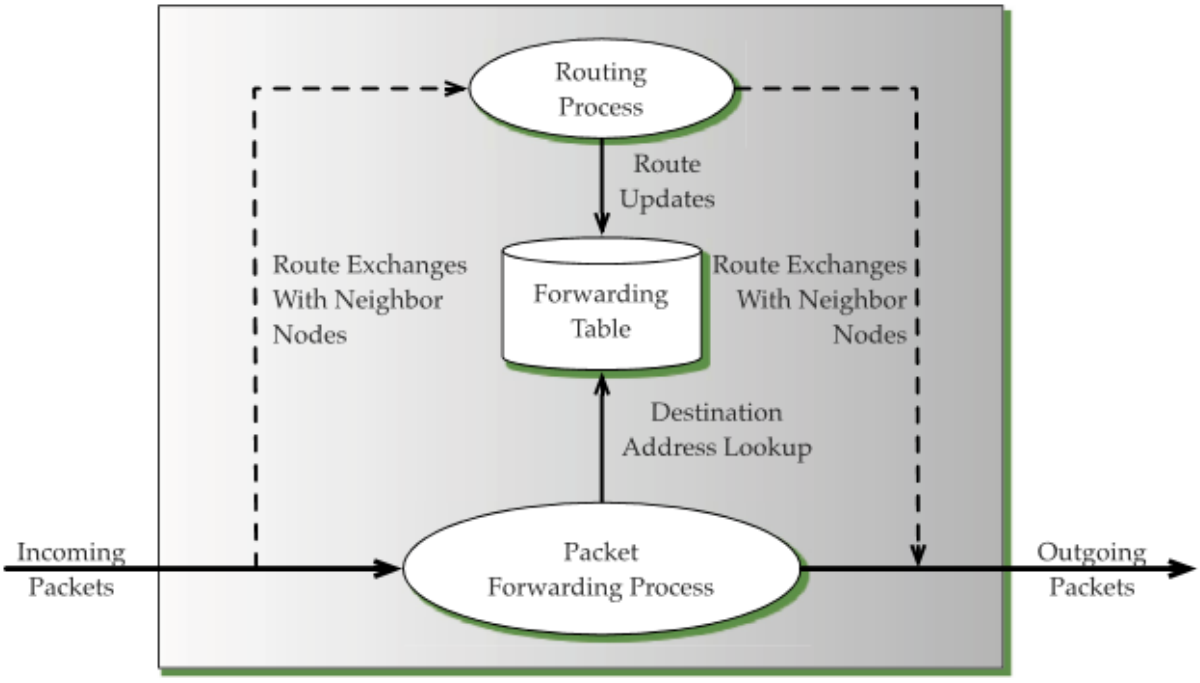


Figure 2.1: Routing and packet forwarding process.

Based on the information exchanged between neighboring routers using routing

protocols, the routing process constructs a view of the network topology and computes the best paths. The network topology reflects the network destinations that can be reached as identified through IP prefix based network address blocks. The best paths are stored in a data structure called the forwarding table. The packet forwarding process moves a packet from an input interface ("ingress") of a router to the appropriate output interface ("egress") based on the information contained in the forwarding table. Since each packet arriving at the router needs to be forwarded, the performance of the forwarding process determines the overall performance of the router.

The functions of the packet forwarding process can be categorized into two subgroups: basic forwarding and complex forwarding. Basic forwarding defines the minimal set of functions a router should implement in order to transfer packets between interfaces. Complex forwarding functions represent the additional processing required by the routers, depending on their deployment environments and their usage.

BASIC FORWARDING FUNCTIONS

For forwarding an IP packet from an incoming interface to an outgoing interface, a router needs to implement the following basic forwarding functions.

- ***IP Header Validation***: Every IP packet arriving at a router needs to be validated. Such a test ensures that only well-formed packets are processed further while the rest are discarded. This test also ensures that the version number of the protocol is correct, the header length of the packet is valid and the computed header checksum of the packet is the same as the value of the checksum field in the packet header.
- ***Packet Lifetime Control***: Routers must decrement the time-to-live (TTL) field in the IP packet header to prevent packets from getting caught in the routing loops forever. If the TTL value is zero or negative, the packet is discarded; an ICMP (Internet Control Message Protocol) message is generated and sent to the original sender.
- ***Checksum Recalculation***: Since the value of the TTL is modified, the header checksum needs to be updated. Instead of computing the entire header

checksum again, it is more efficient to compute it incrementally; after all, the TTL value is always decremented by 1.

- ***Route Lookup***: The destination address of the packet is used to search the forwarding table for determining the output port. The result of this search will indicate whether the packet is destined for the router, to an output port (unicast), or to a set of multiple output ports (multicast).
- ***Fragmentation***: It is possible that the maximum transmission unit (MTU) of the outgoing link is smaller than the size of the packet that needs to be transmitted. This means that the packet would need to be split into multiple fragments before transmission.
- ***Handling IP Options***: The presence of the IP options field indicates that there are special processing needs for the packet at the router. While such packets might arrive frequently, a router nonetheless needs to support those processing needs.

COMPLEX FORWARDING FUNCTIONS

Besides the basic functions, the marketplace has necessitated the need for additional, complex functions. That is, with the popularity of the Internet, complex issues such as security, different user requirements, and service guarantees based on different service level agreements have become paramount and need to be addressed. These issues translate to additional processing when forwarding a packet, without essentially increasing the overall packet processing time at a router. To cite an example of service differentiation, consider a scenario where customers are interested in watching a high definition movie streaming directly over the Internet. Such streaming requires not only high bandwidth but timely delivery of the data. The router needs to distinguish such packets so that it can forward them earlier than packets generated as a result of accessing a website. This results in the notion of differentiated services, and consequently, requires that routers support a variety of mechanisms such as the following:

- ***Packet Classification***: For distinguishing packets, a router might need to examine not only the destination IP address but also other fields such as source address, destination port, and source port. The process of differentiating the packets and applying the necessary actions according to certain rules is known as packet classification.

- ***Packet Translation***: As the public IPv4 address space is being exhausted, there is a need to map several hosts to a single public address. Thus, a router that acts as a gateway to a network needs to support network address translation (NAT). NAT maps a public IP address into a set of private IP addresses and vice versa. This requires a router to maintain a list of connected hosts and their local addresses and to translate the incoming and outgoing packets.
- ***Traffic Prioritization***: A router might need to guarantee a certain quality of service to meet service level agreements. This involves applying different priorities to different customers or data flows and providing a level of performance in accordance with the predetermined service agreements. For example, the agreement might specify that a fixed number of packets have to be delivered at a constant rate, which is necessary for real time streaming multimedia applications such as IPTV, or real-time interactive applications such as VoIP.

BASIC ARCHITECTURE OF A ROUTER

A Generic router consist of the following components:

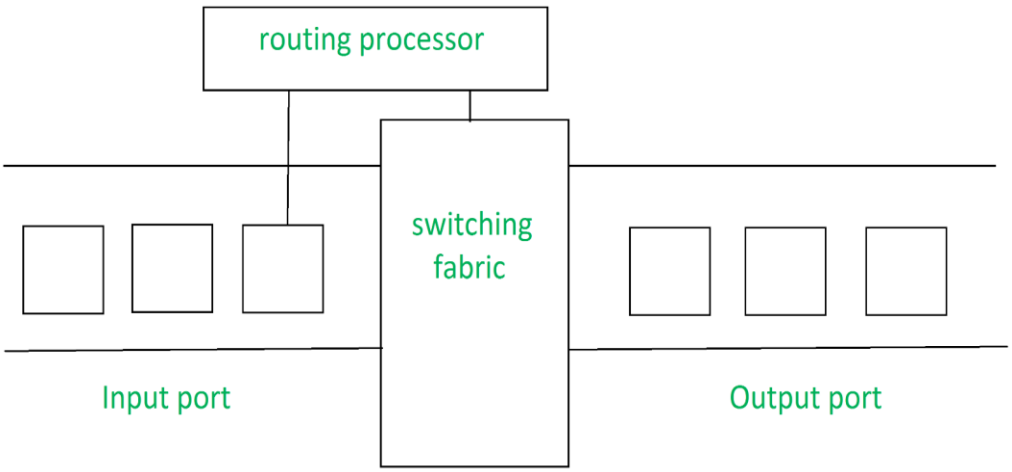
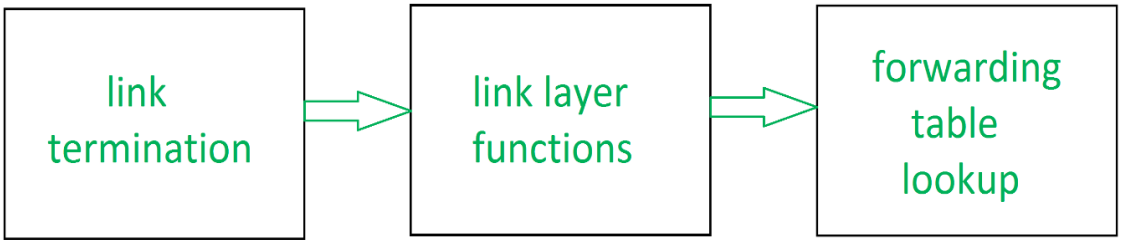


Figure 2.2: Basic Routing architecture.

1. **Input Port –**

   This is the interface by which packets are admitted into the router, it performs several key functions as terminating the physical link at router, this is done by the leftmost part in the below diagram, the middle part does

the work of interoperating with the link layer like decapsulation, in the last part of the input port the forwarding table is looked up and is used to determine the appropriate output port based on the destination address.



Input port

2. **Switching Fabric** –

   This is the heart of the Router; it connects the input ports with the output ports. It is kind of a network inside a networking device. The switching fabric can be implemented in a number of ways some of the prominent ones are:
   1. Switching via memory: In this we have a processor which copies the packet from input ports and sends it to the appropriate output port. It works as a traditional cpu with input and output ports acting as input and output devices
   2. Switching via bus: In this implementation we have a bus which connects all the input ports to all the output ports. On receiving a packet and determining which output port it must be delivered to, the input port puts a particular token on the packet and transfers it to the bus. All output ports are able to see the packets but it will be delivered to the output port whose token has been put in, the token is then scrapped off by that output port and the packet is forwarded
   3. Switching via interconnection network: This is a more sophisticated network, here instead of a single bus we use 2N bus to connect n input ports to n output ports.
3. **Output Port** –
   This is the segment from which packets are transmitted out of the router. The output port looks at its queuing buffers (when more than one packets have to be transmitted through the same output port queuing buffers are formed) and

takes packets, does link layer functions and finally transmits the packets to outgoing link

4. **Routing Processor –**

   It executes the routing protocols; it works like a tradition cpu. It employs various routing algorithm like link-state algorithm, distance-vector algorithm etc. to prepare the forwarding table, which is looked up to determine the forwarding table.

2.2 Routing table versus forwarding table

The packet forwarding function directs an incoming packet to the appropriate output interface based on the results of looking up a forwarding table. The routing function builds a routing table that is used in the construction of the forwarding tables. Often, in the literature, the terms routing table and forwarding table are used interchangeably to refer the data structures in a router for forwarding packets. In this section, we highlight the differences between those tables.

The routing table is constructed by the routing algorithms based on the information exchanged between neighboring routers by the routing protocols. Each entry in the routing table maps an IP prefix to a next-hop. The forwarding table, on the other hand, is consulted by the router to determine the output interface to which an incoming packet needs to be forwarded. Thus, each entry in the forwarding table maps an IP prefix to an outgoing interface. Depending on the implementation, the entries might contain additional information such as the MAC address for the next-hop and statistics about the number of packets forwarded through by using the interface.

While a single table for routing and forwarding is possible, most implementations tend to keep these two separates because of the following reasons. First, the forwarding table is optimized for searching a destination IP address against a set of IP prefixes while the routing table is optimized for calculating changes in the topology. Second, as every packet needs to examine the forwarding table, it is implemented in a specialized hardware for high-speed routers. However, the routing tables are usually implemented in software. An instance of a routing table and forwarding table is shown in Table 13.1.

Table 13.1 Routing table and forwarding table

(a) Routing table

IP prefix	Next hop
10.5.0.0/16	192.168.5.254

(b) Forwarding table

IP prefix	Interface	MAC address
10.5.0.0/16	eth0	00:0F:1F:CC:F3:06

The routing table in the figure indicates the next hop IP address for a destination IP prefix. The forwarding table tells us a packet bound to the network identified by the IP prefix should be forwarded to interface eth0 with the appropriate MAC address.

2.3 TYPES OF ROUTERS

Routers can be of different complexities based on where in the network they are deployed and how much traffic they need to carry. Naturally, this means that routers can be of different types. In this section, we describe three types of routers: core routers, edge routers and enterprise routers and outline their requirements .

Core Routers

Core routers are used by service providers for interconnecting a few thousand small networks so that the cost of moving traffic is shared among a large customer base. Since the traffic arriving at the core router is highly aggregated, it should be capable of handling a large amount of traffic. Hence, the primary requirements for a core router are high speed and reliability. While it is important to keep the cost of a core router reasonable, the cost is a secondary issue.

The speed at which a core router can forward packets is mostly limited by the time spent to lookup a route in the forwarding table. On receiving a packet from an ingress interface, the forwarding ta- ble entries need to be searched to locate the longest prefix match. The prefix represents the target IP network that the packet is destined for. The matching prefix determines the egress interface. With the increase in the number of systems connected to the internet and the associated surge in traffic growth, demand is placed on core routers to forward more packets per second. Hence, specialized algorithms implemented in hardware are required for fast and efficient lookups.

Since core routers form the critical nodes in the network, it is essential that these routers do not fail under any conditions. The reliability of a router depends upon the reliability of physical elements such as the line cards, switch fabric, and route control processor cards. The reliability of these physical elements is achieved by full redundancy – dual power supplies, standby switch fabric, duplicate line cards, and route control processor cards. Moreover, the software is enhanced so that when one of the elements fails, the packet forwarding and the routing protocols continue to function.

Edge Routers

Edge routers, also known as access routers, are deployed at the edge of the service provider networks for providing connectivity to customers from home and small businesses. The first generation of edge routers were really remote access servers attached to terminal concentrators that aggregated a large number of slow-speed dial-up customers. However, this is not the case anymore. First, the need for more bandwidth has led to the introduction of a variety of access technologies such as high-speed modems, DSL, and cable modems. Hence, the edge routers need to support aggregation of customers using different access technologies. Second, in addition to legacy remote access protocols, these routers need to implement newer protocols such as the point-to-point tunneling protocol (PPTP) and IPsec that support VPNs. These protocol implementations should also scale as they need to be run on every port.

Finally, these routers should be capable of handling large amounts of traffic. This is necessary as many customers are migrating from dial up access to high-speed modems. These trends suggest that the edge routers support a large number of ports capable of different access technologies and many protocols operating at each port.

Enterprise Routers

Enterprise networks interconnect end systems located in companies, universities, and so on. The primary requirements of routers in these networks are to provide connectivity at a very low cost to a large number of end systems. In addition, a desirable requirement is to allow service differentiation to provide QoS guarantees for different departments of an enterprise.

A typical enterprise network is built using many Ethernet segments, which are interconnected by hubs, bridges, and switches. These devices are inexpensive and can be easily installed with limited configuration effort. A network built using such inexpensive devices tends to degrade in performance as the size of the network increases. Hence, using routers in these networks to divide the end systems into hierarchical IP subnetworks, is desirable. Moreover, it scales the network better.

In addition to providing the basic connectivity, there are several additional design requirements for the enterprise routers. First, these routers require efficient support for multicast and broadcast traffic as applications such as video broadcasting are more predominantly used in the enterprise. Second, these routers need to implement many legacy technologies, which are still in use in the enterprises.

The third requirement is the extensive support for security firewalls, filters, and VLANs. Finally, as these routers have to connect many LANs, it is required that they support large number of ports.

For enterprises, the network is considered as an operational expense and the goal is to minimize this expense. Hence, the routers targeted for enterprise deployment are required to have low per port cost, a large number of ports and ease of maintenance. Hence, it is challenging to design an enterprise router that satisfies these requirements for every port and still keep the cost per port low.

2.4 ELEMENTS OF ROUTERS

So far, we have discussed the functions and types of a router; we next discuss the elements needed in a router to provide these functions. For this purpose, a router can be viewed from two different perspectives. From a functional perspective, it can be logically viewed as a collection of modules where each module implements a set of related functions to achieve the overall goal of forwarding packets.

From an architectural perspective, a router can be considered as an interconnection of different types of cards running specialized software. We discuss the functional perspective before examining the architectural perspective. A router can be divided into several modules from a functional point of view. These components implement the various requirements of a router described in the previous sections.

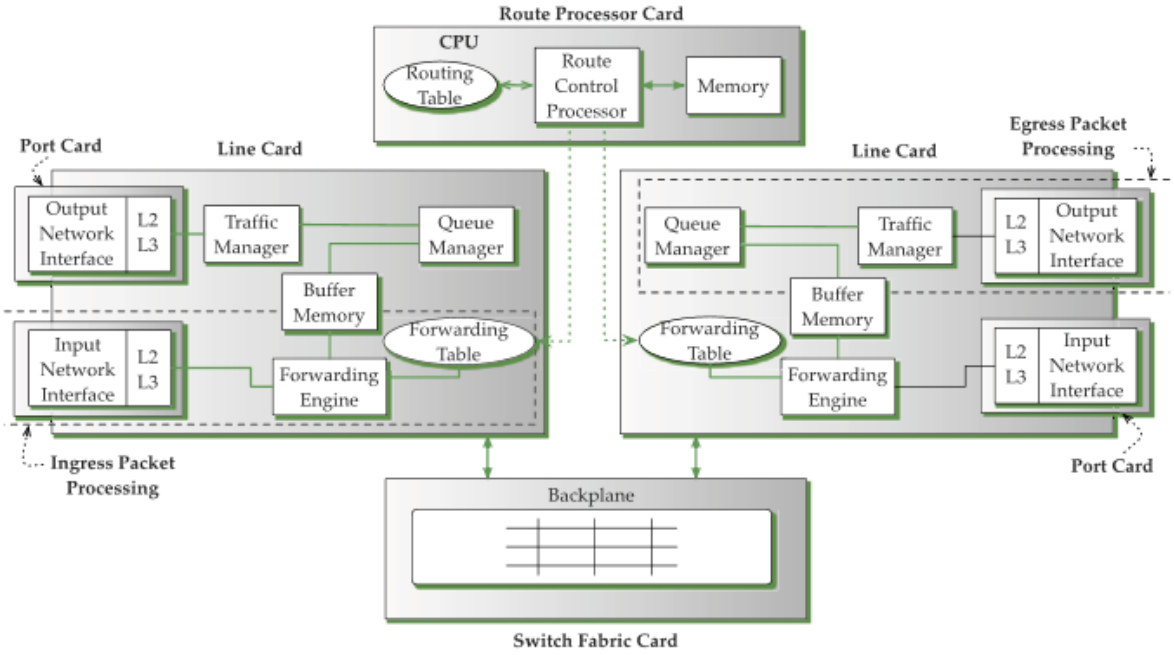


Figure 2.3: Components of Router.

A generic router consists of six major functional modules: network interfaces, a forwarding engine, a queue manager, a traffic manager, a backplane, and a route control processor. These functional modules are shown in Figure 2.3.

- ***Network Interfaces***: A network interface contains many ports that provide the connectivity to physical network links. A port terminates a physical link at the router and serves as the entry and exit point for incoming and outgoing packets, respectively. A port is specific to a particular type of network physical media. For instance, a port can be an Ethernet port or a SONET interface. In addition, a network interface provides several functions. First, it understands various data link protocols so that when the packet arrives it can decapsulate the incoming packets by stripping the Layer 2 (L2) headers. Second, it extracts the IP headers, i.e., the Layer 3 (L3) headers, and sends them to the forwarding engine for route lookup while the entire packet is stored in memory. Collectively, this processing is referred to as L2/L3 processing. Further, it provides the functionality of encapsulating L2 headers before the packet is sent out on the link.

- ***Forwarding Engines***: These are responsible for deciding which network interface the incoming packet should be forwarded to. When a port receives a new packet, it de-encapsulates L2 headers and sends the entire IP packet, or just the packet header, to the forwarding engine. The forwarding engine consults a table, i.e., engages in a route lookup function, and determines which network interface the packet should be forwarded to. This table is called the forwarding information base or simply the forwarding table. Algorithms for route lookups can be implemented in custom hardware or software running on a commodity hardware. Depending on the architecture, the lookups can occur in the custom hardware or in a local route cache in the line card. In order to provide QoS guarantees, forwarding engines may need to classify packets into predefined service classes.
- ***Queue Manager***: This component provides buffers for temporary storage of packets when an outgoing link from a router is overbooked. When these buffer queues overflow due to congestion in the network, the queue manager selectively drops packets. Thus, the responsibility of this component is to manage the occupancy of the queue and implement policies about which packets to drop when the queues are about to be fully occupied.
- ***Traffic Manager***: This component is responsible for prioritizing and regulating the outgoing traffic, depending upon the desired level of service. This is necessary as routers carry traffic from different subscribers, and it is important to ensure that they get the level of service for which they pay. The traffic manager shapes the outgoing traffic to the subscriber according to the service level agreement. Similarly, when a router receives traffic from a subscriber, the traffic manager ensures that it does not accept more than what is specified in the contract. Sometimes, the functionality of the queue manager and the traffic manager are merged into a single component.
- ***Backplane***: This component provides connectivity for the network interfaces so that packets from an incoming network interface can be transferred to the outgoing network interface card. The backplane can either be shared, where only two interfaces can communicate at any instant or be switched, where multiple interfaces can communicate simultaneously. The aggregate bandwidth of all the attached network interfaces defines the bandwidth required for the backplane.

- ***Route Control Processor***: The control processor is responsible for implementing and executing routing protocols. It maintains a routing table that is updated whenever a route change occurs. Based on the contents of the routing table, the forwarding table is computed and updated. In addition, it also runs the software to configure and manage the router. A route control processor also performs complex packet-by-packet operations like errors during packet processing. For example, it handles any packet whose destination address cannot be found in the forwarding table in the line card b sending an ICMP packet to its source of origin indicating the error. Due to the variety of different tasks, these functionalities are typically implemented in software running on a general-purpose microprocessor.

We consider the architectural perspective and how the functional modules are implemented in practice. Various architectural components of a router are as follows:

- ***Port Cards:*** A port card implements the network interfaces. Each port card is capable of handling only a specific media. For instance, a port card will support only Ethernet while the other can handle only SONET. The port cards contain an L2 processing logic that understands the L2 packet format specific for that media. In addition, the port cards perform accounting about the incoming and outgoing packets. Such cards are given different names by different vendors; for example, Juniper Networks refers to them as Physical Interface Cards (PICs) while Cisco refers to them as Physical ayer Interface modules (PLIMs) in CRS-1 routers.
- ***Line Cards:*** A line card implements a majority of the functional components, forwarding engine, queue manager, and traffic manager. It parses the IP payload and uses the contents of the header to make decisions about forwarding, queueing, and discarding during periods of link congestion. It also contains memory buffers for storing packets during processing and queueing. The line card houses the port cards, connects to the backplane, and ultimately to another line card. Sometimes, the line cards themselves implement the ports specific to certain media rather than using port cards.
- ***Switch Fabric Cards:*** While a line card implements the packet processing functions, a switch fabric card serves as the backplane for transferring packets

from an ingress line card to an egress line card. In high end routers, multiple switch fabric cards are used for increased throughput and redundancy.

- ***Route Processor Cards***: These cards implement the functionality of the route control processor. The routing protocols and the management software run on these cards. In high end routers, these cards use general purpose processors with a large amount of memory running a commodity operating system.

2.5 PACKET FLOW

The packet flow in a generic router is shown in Figure 2.4. The processing steps can be broadly grouped into ingress packet processing and egress packet processing.

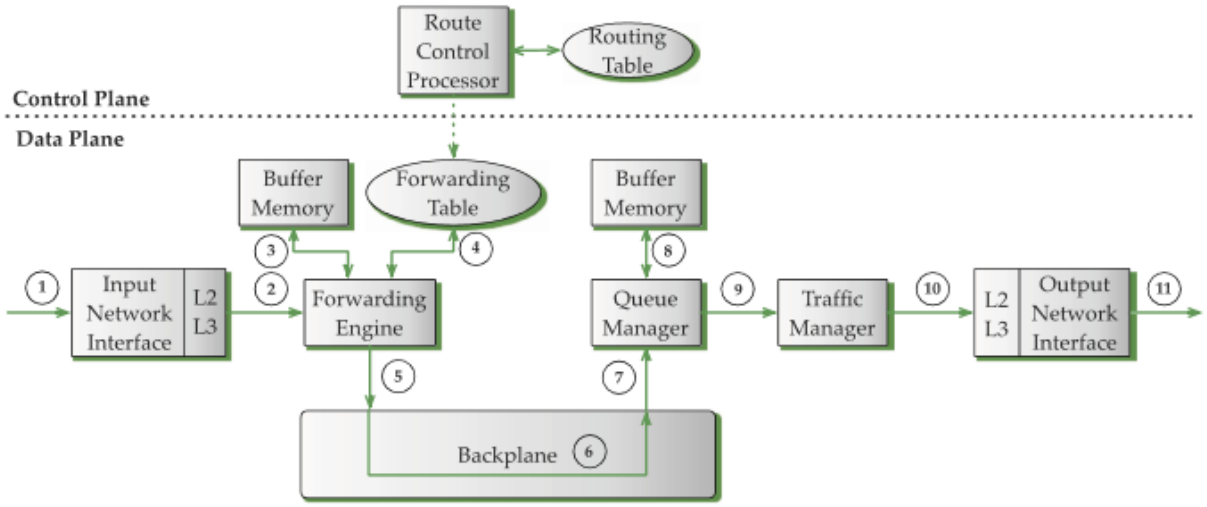


Figure 2.4: Packet Flow in a Router.

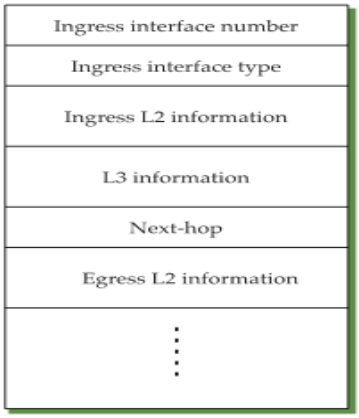


Figure 2.5 Typical fields of a packet context.

INGRESS PACKET PROCESSING

When an IP packet arrives from the network, it first enters the network interface. For the sake of discussion, let us assume that the packet is received on an Ethernet port. The network interface interprets the Ethernet header, detects frame boundaries, and identifies the starting points of the payload and the IP packet in the frame. The L2 processing logic in the card removes the L2 header and constructs a packet context. A packet context is a data structure that essentially serves as a scratch pad for carrying information between different stages of packet processing inside the router. The L2 processing logic appends to the packet context information about L2 headers, for instance, in the case of Ethernet, this would be the source and destination MAC address. In addition to L2 information, the packet context can carry additional information which are shown in Figure 2.5. Use of other fields in the packet context will be revealed later in the discussion.

Now the L2 processing logic peels off the payload, which is an IP packet, and along with the packet context sends it to the L3 processing logic. The L3 processing logic locates the IP header and checks its validity. It extracts the relevant IP header information and stores it in the packet context. The header information includes the destination address, source address, protocol type, DSCP bits (for differentiated services) and if the IP packet is carrying TCP or UDP payload, the destination and the source ports as well.

At this point, the packet context contains enough information for route lookup and classification of the packet. Next, the entire packet context is sent to the forwarding engine in the line card. The forwarding engine searches a table (the forwarding table) to determine the next-hop. The next-hop information contains the egress line card and the outgoing port the packet needs to be transferred. This information is populated in the packet context.

While the forwarding engine is determining the next-hop using the packet context, the L3 processing logic sends the IP packet to be temporarily stored in the buffer memory. When the forwarding engine completes its part, the packet context is appended with the address of the packet in memory and it is sent to the backplane interface. From the packet context, the backplane interface knows to which line card the packet needs to be transferred. It then schedules the packet for transmission along

with the packet context over the backplane. Note that the priority of the packet is taken into account while transmitting on the backplane: higher priority packets need to be scheduled ahead of lower priority packets.

EGRESS PACKET PROCESSING

When the packet reaches the egress line card, the backplane interface on the egress line card receives the packet and stores it in the line card memory. Meanwhile, the received packet context is updated with the new address of the memory location and sent to the queue manager. The queue manager examines the packet context to determine the packet priority. Recall that the priority was determined by the forwarding engine in the ingress line card during packet classification. Next the queue manager inserts the context of the packet in the appropriate queue.

As different queues, depending on the priority, consume different amounts of bandwidth on the same output link, the queue manager implements a scheduling algorithm. The scheduling algorithm chooses the next packet to be transmitted according to the bandwidth configured for each queue. In some instances, the queues could be full because of congestion in the network. In order to handle such cases, the queue manager implements packet dropping behavior to proactively drop packets when the router experiences congestion.

Once the packet is scheduled to be transmitted, the traffic manager examines its context to identify the customer and to see if there are any transmit rate limitations that need to be enforced according to the service contract. Such a mechanism is referred to as traffic shaping. If the traffic exceeds any rate limitations, the traffic manager delays or drops the packet in order to comply with the agreed rate.

Finally, the packet arrives at the network interface where L3 processing logic updates its TTL and updates the checksum. The L2 processing logic, in function of the physical media, adds the appropriate L2 headers, and the packet is transmitted on the appropriate port.

2.6 PACKET PROCESSING

The tasks performed by a router can be categorized into time critical and non-time critical operations depending on their frequency; they are referred to as fast path and slow path, respectively. The time critical operations are those that affect the majority of the packets and need to be highly optimized I order to achieve gigabit forwarding rates. The time critical tasks can be broadly grouped into header processing and forwarding. The header processing functions include packet validation, packet lifetime control and checksum calculation, while the forwarding functions include destination address lookup, packet classification for service differentiation, and packet buffering and scheduling. Since these tasks need to be executed for every packet in real time, a high-performance router implements these fast path functions in hardware.

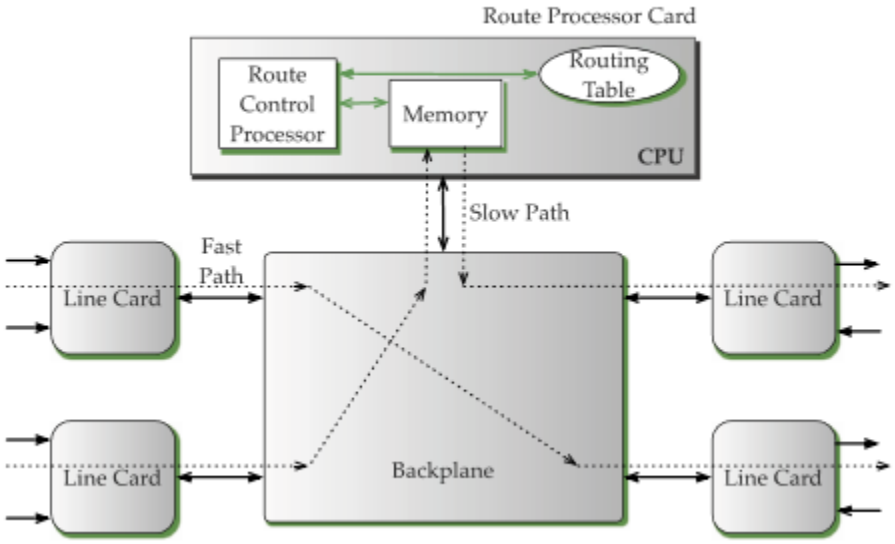


Figure 2.6: Router Component.

Non-time critical tasks are typically performed on packets destined to a router for maintenance, management, and error handling. Such tasks include, but are not limited to the following:

- Processing of data packets that lead to errors in the fast path and generation of ICMP packets to inform the originating source of the packets.
- Processing of routing protocol keep-alive messages from adjacent neighbors and sending these messages to the neighboring routers.
- Processing of incoming packets that carry route table updates and sending messages to neighboring routers when network topology changes.
- Processing of packets pertaining to management protocols, such as SNMP, and the associated replies

These slow path tasks are integrated so that they do not interfere with the fast path mechanism. In other words, time critical operations must have the highest priority under any circumstances. The fast path and slow path are identified in Figure 2.6. As shown in the figure, a packet using the fast path is processed only by the modules in the line cards as it traverses the router. On the other hand, a packet on the slow path is forwarded to the CPU, as many of the slow path tasks are implemented by the software running on it. Such an implementation is advantageous, as there is a clear separation between the fast path and slow path. Consequently, there is no interference with the performance of packets on the fast path.

In the figure, the route processor card is directly attached to the backplane, as are the line cards. The packets on the slow path from the line cards are forwarded through this link to the CPU in the router processor card. The CPU dispatches them to the appropriate protocol handlers for processing.

Similarly, the protocols running on the CPU can generate IP packets to be transmitted to the network. These are forwarded to the appropriate line card, as if they were from another line card. For these packets, the CPU needs to perform the route lookup; otherwise, it cannot deduce which line card the packet needs to be forwarded to. Therefore, the CPU needs to maintain its own routing table.

Alternatively, instead of using a separate routing table, it can consult the master routing table of the router that it maintains. Having delineated the distinction between the fast path and slow path, the next logical question is: which router functions need to be implemented on the slow path, and which need to be implemented on the fast path? Many of the forwarding functions are typically implemented in the fast path, while the routing protocol and management functions

are implemented in the slow path. However, for certain functions, it is not obvious how they should be implemented. In the following two sections, we will study in detail some of the fast path and slow path functions.

FAST PATH FUNCTIONS

In the fast path, the packets are processed and transferred from the ingress line card to the egress line card through the backplane. To achieve high speeds, the fast path functions are implemented in custom hardware, such as ASICs. While such custom implementations are less flexible, the increasing need for more packet processing at the router, and the relatively small changes in the IP packet format, make the custom hardware implementation attractive.

Now let us examine some of the fast path operations in detail.

IP Header Processing

As soon as an IP packet enters a router, it is subjected to a set of validity checks to ensure that the packet is properly formed and the header is meaningful. Only well-formed packets can be further processed, otherwise the packet is discarded.

The processing begins with a verification of the protocol version, as routers can support either IPv4or both IPv4 and IPv6. If the version number does not match, then the packet could be malformed.

The second step is for the router to check whether the length of the packet reported by the MAC or the link layer is at least the minimum legal length of an IP packet. This test ensures that the IP header is not truncated by the MAC layer and filters packets less than the minimum intended length. Next, for IPv4, the value of the IP header checksum must equal the calculated header checksum computed by the router.

The routers must decrement the time-to-live field in the IP header to prevent packets from getting caught in routing loops forever and consuming network resources. A packet destined for the local address of the router will be accepted by the router if it has zero or a positive value of TTL. On the other hand, the packets that are being forwarded by the router should have their TTL value decremented and checked whether the TTL value is positive, zero or negative. A positive value of TTL

indicates that the packets have more life left and such packets are actually forwarded. The remaining packets that have a TTL value equal to or less than zero are discarded and an ICMP error message is sent to the original sender.

Since the TTL field has been modified, the IP header checksum must be recalculated. A naive approach is to compute the checksum over the entire IP packet again, which could be computationally expensive.

Packet Forwarding

The function of packet forwarding is to determine on which network interface a packet needs to be transmitted out of the router. The forwarding engine module controls this function using a forwarding table. The forwarding table is a summary of the routing table created by the route control processor. The router extracts the destination IP address from an incoming packet and performs a lookup in the forwarding table to determine the next hop IP address for the packet. This procedure also decides which output port and network interface should be used to send the packet. The result of the lookup could lead to three possibilities:

- ***Local:*** If the IP packet is destined for the router's local IP addresses, it is delivered to the route control processor. For example, the destination for the packets carrying routing protocol keep-a lives and route updates is the router itself.
- ***Unicast***: The packet needs to be delivered to a single output port on a network interface, either to a next-hop router or to the ultimate destination. In the latter case, the router is directly connected to the destination network.
- ***Multicast***: The IP packet is delivered to a set of output ports on the same or different network interfaces, based on multicast group membership, which is maintained by the router.

As the volume of data traffic grows, routers are expected to forward more packets per second.

Packet Classification

In addition to forwarding packets, the routers need to isolate different classes, or types, of IP traffic, based on information carried in the packet. Subsequently,

depending on the type of IP traffic, the appropriate action is applied. This process of selectively identifying packets and applying the necessary actions according to certain rules is known as packet classification. A set of such rules are referred to as a classifier. A router should be capable of discriminating packets not only with the destination address, but also with the source address, source port, destination port, and protocol flags, commonly referred to as a 5-tuple. The source and destination address identify the participating endpoints, the protocol flags identify the type of payload, and the source and destination ports identify the application (assuming the payload is TCP or UDP). The packet classification function should be fast enough to keep up with the line rate. Hence, the algorithms for classification need to be fast and efficient.

Packet Queueing and Scheduling

As routers keep forwarding packets, there can be an instance where multiple packets arriving on different ingress network interfaces need to be forwarded to the same egress network interface simultaneously. Such burstiness in the Internet traffic requires buffers that serve as a temporary waiting area for packets to queue up before transmission. The order in which they are transmitted is determined by various factors such as the service class of the packet, the service guarantees associated with the class, etc.

Therefore, routers not only provide buffers but also require sophisticated scheduling functions. The scheduling function prioritizes the traffic based on the bandwidth requirements and tolerable amount of delay by choosing the appropriate packet from these buffers. Without such options, packets simply line up and are transmitted in the order in which they are received (FIFO). While many data applications like file transfers and web browsing can tolerate some delay, for delay-sensitive applications such as VoIP, FIFO behavior is not clearly desirable. Chapter 17 discusses in detail various scheduling algorithms and their advantages and disadvantages.

When a network is congested, traffic arriving at the router could fill up its buffers, thereby dropping subsequent packets. If congestion could be detected before it actually occurs, proactive measures can be taken for prevention. Some of these measures include packet dropping when the occupancy of buffers reaches a

predefined threshold. As the packet dropping function needs to determine whether a packet needs to be dropped, it is considered as a fast path function.

SLOW PATH OPERATIONS

The packets following the slow path are partially processed by the ingress line card before forwarded to the CPU for further processing. Once the CPU completes processing, it directly sends those packets to the egress line card. For the next few sections, we shall study some of the slow path functions in detail.

Address Resolution Protocol (ARP)Processing

When a packet needs to be sent on an egress interface, the router needs to determine the data link or the MAC address for the destination IP address or the next-hop IP address. This is because the network interface hardware on the router, to which the packet needs to be forwarded understands only the addressing scheme of that physical network. Hence, a mechanism is needed to translate the IP address to a link-level address (for Ethernet, it is a 48-bit MAC address). Once the link-level address is determined, the IP packet can be encapsulated in a frame that contains the link-level address an transmitted either to the ultimate destination or to the next-hop router.

A router that forwards IP packets to the destination address must either maintain these link-level addresses or dynamically discover them. The mechanism for discovering them dynamically requires the use of address resolution protocol (ARP). ARP assumes that the underlying network supports link level broadcasts and sends a query ARP request containing the IP address. When the ARP reply comes in from the host with the link-level address, it is maintained as a part of the forwarding table in the router. These entries are timed out periodically and removed and rediscovered again since the mappings change over time (possibly, the media card could have been changed).

When a packet needs to be forwarded, these link-level addresses are obtained as a result of the address lookup operation on the forwarding table along with the outgoing interface. Hence, a router designer might choose to implement ARP processing in the fast path for two reasons: performance and the need for direct access to the physical network. Other designers might choose to implement ARP in

the slow path, since it does not occur very frequently. When implemented in the slow path, an IP packet arriving in the router whose link-level address is not known, is forwarded to the central CPU. The CPU initiates an ARP request and once the ARP reply arrives, the IP packet is forwarded. The CPU updates the forwarding tables in the line cards with the link-address for future packets.

Fragmentation and Reassembly

Since a router connects disparate physical networks, there can be scenarios in which the maximum transmission unit (MTU) of one physical network is different from the other. When this happens, an incoming IP packet can be fragmented into small packets by the router if the output port is incapable of carrying the packet with its original length, i.e., the MTU of the output port is less than that of the input port. Thus, fragmentation enables transparent connectivity even across physical networks with different MTU sizes. However, the downside of fragmentation is that it adds more complexity in the design of the router and reduces the overall data throughput since the entire IP packet needs to retransmitted even if a fragment is lost.

As the fast path is implemented in hardware in high-speed routers, adding support for fragmentation in hardware could be complex and expensive. The need to fragment packets is often an exceptional condition, since a large percentage of packets are related to web traffic (the average size is approximately 250 bytes) and TCP ACKs (40 bytes). When path MTU discovery is used, meaning that the smallest MTU size in a path is discovered apriori packet transmission, the need for fragmentation is very rare. Therefore, fragmentation is usually implemented in the slow path.

For further efficiency, fragmented packets transiting through a router are not reassembled, even if the output port is capable of supporting a higher MTU. The rationale is that it makes the design of the router complex, especially in the fast path, and the end system will be capable of reassembling it anyway. Implementing reassembly in the fast path requires handling of packets arriving out of order, detecting lost fragments and discarding the remaining fragments in the buffers. Such tasks are complex to implement in hardware. However, packets destined for the router should be reassembled and usually it is implemented in software. Fragment reassembly can consume substantial amounts of both CPU and memory resources.

The percentage of packets sent to the router is normally quite low relative to the packets transiting through the router, which is another argument for fragmentation to be implemented in the slow path.

Advanced IP Processing

Some of the advanced IP options include source routing, route recording, time stamping and ICMP error generation. Source routing allows the sender of a packet to specify the route it should take to reach the destination. The main arguments for implementing these functions in the slow path is that the packets requiring these functions are rare and can be handled as exceptional conditions. Hence, these packets can be processed in the control processor in the slow path.

For reporting errors about IP packets with invalid headers, the control processor can instruct the ingress network interface to discard the packet. Another alternative is to discard the packet in the fast path and send a notification to the control processor that generates an ICMP message. Some designers consider that it is more efficient to store templates of various errors in the forwarding engine, and then combine them with the IP header of the invalid packet to generate a valid ICMP message immediately.

2.7 New Router Architectures

A new classification of router architectures that differs from the traditional classification. Our classification scheme is based on how the packet forwarding function is implemented from the view point of a line card.

In the new scheme, the router architectures are broadly classified into the following:

- Shared CPU architectures,
- Shared forwarding engine architectures,
- Shared nothing architectures,
- Clustered architectures.

Furthermore, each of this architecture can be considered as an instance of mapping various routing functional modules to architectural components.

2.7.1 SHARED CPU ARCHITECTURES

This architecture is built around a conventional computer architecture; a CPU with memory and multiple line cards are connected by a shared backplane. Each line card implements a network interface to provide connectivity to the external links. The CPU runs a commodity real time operating system and implements the functional modules, including the forwarding engine, the queue manager, the traffic manager, and some parts of the network interface, especially L2/L3 processing logic in software. In addition, the same CPU also incorporates the functionality of the route control processor that implements the routing protocols, route table maintenance, and router management functions. All the line cards share the CPU for their forwarding function; hence, the name shared CPU architecture.

An instance of this architecture is illustrated in Figure 13.6. Note that the figure also captures the flow of a packet and each step is indicated by a number enclosed in a circle. When a packet arrives at the line card, it raises an interrupt to the CPU. The interrupt service routine schedules a transfer of the packet to the buffer memory through the shared backplane. Once the transfer is complete, the CPU extracts the headers of the packet and uses the forwarding table to determine the egress line card and the outgoing port. The packet is subsequently prioritized by the queue manager and shaped by the traffic manager. Finally, the packet is transferred from the memory to the appropriate output port in the egress line card. As one can see, each packet is transferred twice over the shared backplane; once from the ingress line card to the shared CPU and once from the shared CPU to the egress line card.

While most cycles of the CPU are used for packet forwarding, it spares some of its cycles running the routing protocols. It periodically exchanges protocol keep-alive messages with the neighbor routers; whenever a route change occurs it incrementally updates the routing table and the forwarding table. In addition, the CPU also executes management functions for configuring and administering the router.

A significant design issue, in this architecture, is how the CPU divides its execution cycles between the control path and data path software. Meanwhile, the shared CPU spends some of its cycles running the routing protocols. It periodically exchanges protocol keep-alive messages with the neighbor routers; whenever a route change occurs it incrementally updates the routing table and the forwarding table.

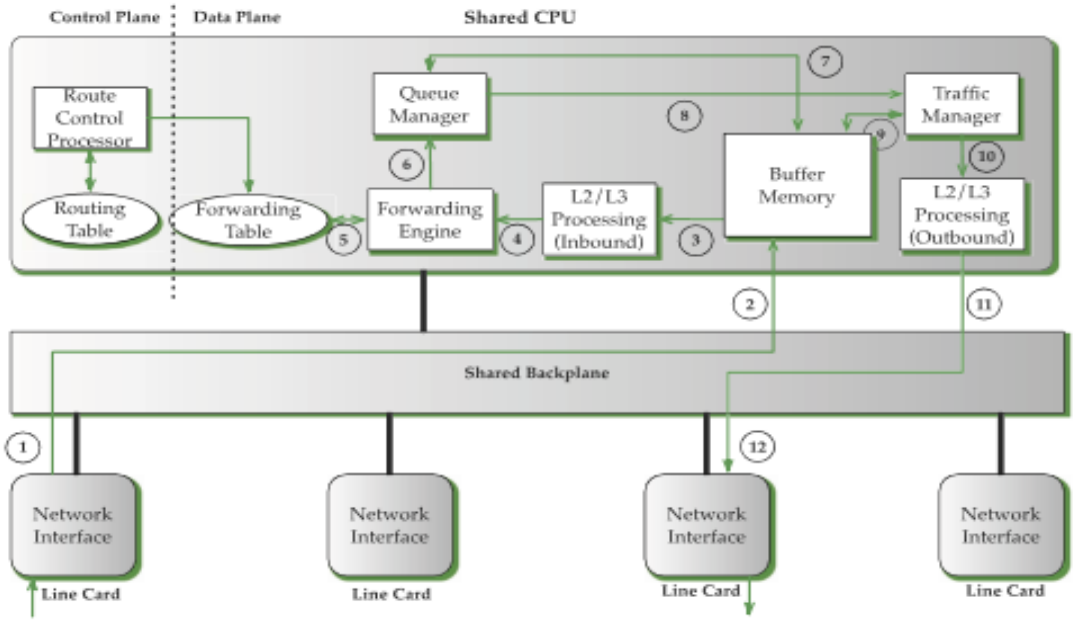


Figure 2.7: Shared CPU architecture.

The main advantages of this architecture are the simplicity and the flexibility of implementation.

However, the following bottlenecks present in the system limit the performance of this architecture.

- Each packet entering the system has to traverse the CPU; thus, the limited number of CPU cycles results in a processing bottleneck.
- The packet forwarding functions, such as the forwarding table lookup, buffering and retrieval of the packet involve accessing memory. Due to mismatch in speed between the memory and CPU, access to memory contributes to a large amount of overhead. The memory access speeds have increased little over the last few years.
- The shared backplane becomes a severe limiting factor as each packet has to traverse the backplane twice. This effectively reduces the throughput by a factor of two.

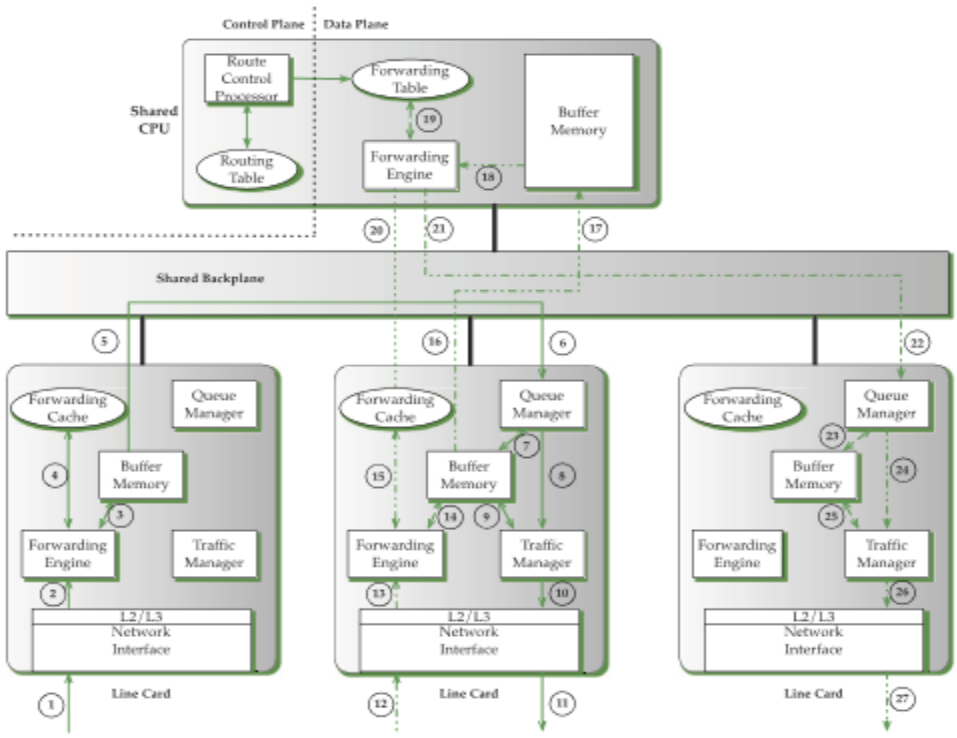


Figure 2.8: Shared CPU architecture with route caches.

To summarize, the performance of this architecture depends heavily on the throughput of the shared backplane, the speed of the shared CPU and the cost of memory access. Hence, this architecture does not scale well to meet increasing throughput requirements. However, for low end access and enterprise routers, where the throughput requirements are less than 1 Gbps, this architecture is still used.

Assuming the CPU speed and the cost of memory access remains the same, the throughput of the shared CPU architecture can be increased if the packet traverses the shared backplane once instead of twice. If the functionality of the forwarding engine can be offloaded to the line cards, the packets need to be transferred through the backplane only once (just to the egress line card). Such an architecture is shown in Figure 2.8. The basic idea is that caching the results of the route lookup in the line card allows many of the incoming packets to be transferred directly to the egress line card; thus, increasing the throughput.

As shown in the figure, this architecture also consists of a CPU with buffer memory and line cards connected to a shared backplane. Unlike the previous architecture, more intelligence is added to the line cards, with processor, memory, and forwarding

caches. The CPU maintains the central forwarding table and the line cards cache a subset of the master forwarding table based on recently used routes.

When a first packet from a new destination arrives in the line card, it is sent to the shared CPU, which looks up its route using the central forwarding table. The result of the lookup is then added to the forwarding cache in the ingress line card. This allows subsequent packet flows to the same destination to match the cached route entry, and the packet is directly transferred to the egress line card.

Figure 2.8 identifies the flow of two different packets. The first, indicated by steps 1 through 11, shows the case when the destination address of the incoming packet is found in the forwarding cache. The second identifies the case when the search for the destination address fails in the forwarding cache and the central forwarding table in the CPU needs to be consulted, which is indicated by steps 12 through 27. Since cache memory is limited, the entries are discarded based on LRU (least recently used) or FIFO (first-in first-out) to make space for new entries. The cache entries are periodically aged out to keep the forwarding cache current and, in the case of a route change, immediately invalidated.

The advantage of this architecture is the increased throughput because the forwarding cache of frequently seen addresses in the line card allows it to process packets locally some of the time. However, the throughput is, in fact, highly dependent on the incoming traffic.

2.7.2 SHARED FORWARDING ENGINE ARCHITECTURES

In the shared CPU architecture, we identified that the shared CPU is one of the major bottlenecks, as it is in the path of every packet flow. This architecture is an attempt to mitigate the bottleneck by offloading the functionality of the forwarding engine to a dedicated card called forwarding engine cards. Each forwarding engine card contains a dedicated processor executing the software for route lookup and memory for storing the forwarding table. With multiple such cards, many packets can be processed in parallel, which considerably scales the packet forwarding speed.

In this architecture, multiple line cards are connected through a shared backplane through which the packets are transferred from one line card to another. Line cards and forwarding engine cards are connected together through a separate shared

backplane called the forwarding backplane. The rationale behind using two different backplanes is to keep the data traffic separate from the traffic generated for the forwarding engine cards, thereby improving throughput. This architecture is shown in Figure 2.9, which also illustrates the packet flow. In the figure, the numbers enclosed in circles indicate the steps of packet processing in order.

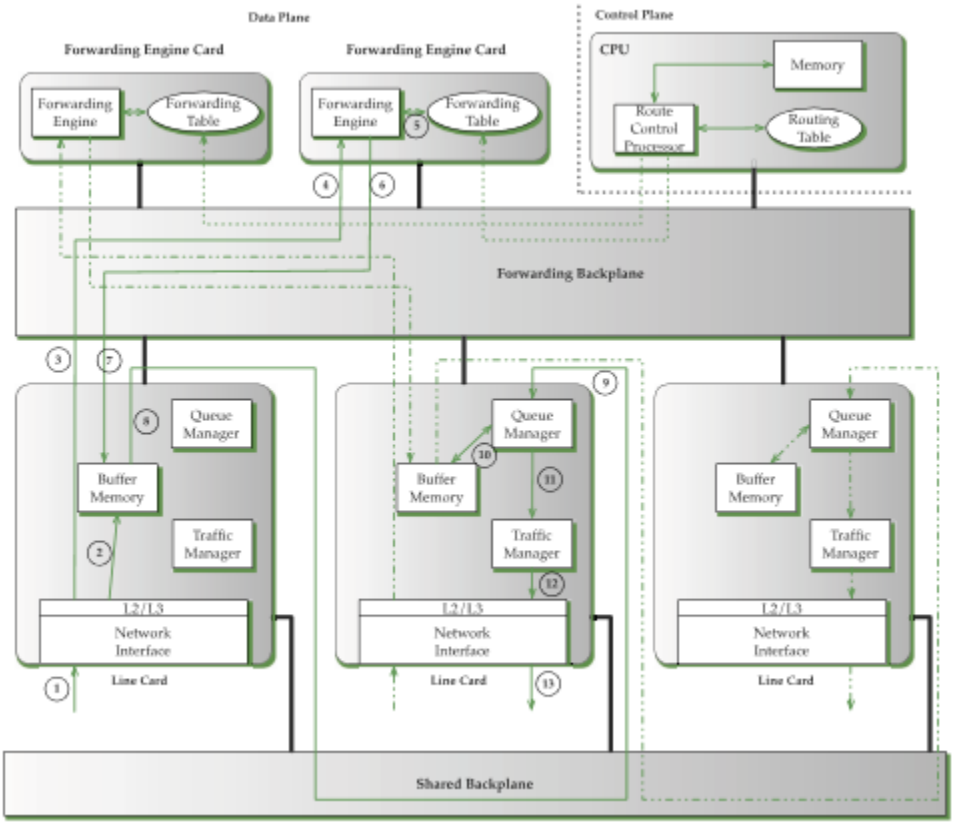


Figure 2.9: Shared forwarding engine architecture using two shared backplanes..

When the ingress line card receives packets, the IP header is stripped and augmented with the packet context containing an identifying tag. The packet context and the IP header are sent to a forwarding engine through the forwarding backplane for IP header validation and route lookup. Since the forwarding engine is responsible for route lookup, just sending only IP headers eliminates the unnecessary overhead of transferring the packet payload over the forwarding backplane.

While the forwarding engine is performing the lookup, the packet payload is buffered in the memory of the ingress line card. The result of the route lookup determines the egress line card and the interface where the packet needs to be

transmitted. This information is stored in the packet context that is followed by decrementing the TTL and updating the checksum in the IP header. The updated header along with the packet context containing the tag is sent to the ingress line card. Upon examining the packet context, the ingress line card transfers the packet from its buffer memory through the shared backplane to the egress line card. Subsequently, it is queued in the buffer memory of the egress line card until the queue manager and traffic manager decide to transmit on the outgoing link.

The route control processor maintains the routing table by exchanging route update messages and computes the forwarding table. The forwarding table is propa gated to all the forwarding engines when a new route is added or an existing route is updated or deleted. Since the forwarding tables at the forwarding engines have the same contents, their consistency needs to be maintained.

Since there are multiple forwarding engines, multiple IP headers can be processed in parallel. This could lead to the situation where packets that arrived later might finish their route lookup earlier than the packets that entered the router earlier. Subsequently, these packets will depart from the router earlier causing packet reordering. The routers need to maintain packet ordering since sequencing of packets in a TCP connection needs to be maintained. If not, it can trigger retransmits, thereby reducing throughput leading to degrading the performance of the overall network.

The L2/L3 packet processing logic in the ingress line card removes the IP headers and assigns these headers to forwarding engines in a round robin fashion. To ensure packet ordering, the packet processing logic in the egress interface also follows round robin, guaranteeing that packets are sent out in the order in which they are received.

The main advantage of this architecture is the ability to scale to higher forwarding speeds. Another advantage of this architecture is that it provides flexibility; the forwarding engine cards can be added whenever needed so that the necessary forwarding speed can be achieved for high-speed core routers.

A key drawback is the use of a shared backplane that does not provide sufficient bandwidth for transmitting packets between line cards and limits the router throughput. Hence, in order to remove this bandwidth limitation, the shared backplane is replaced by a switched backplane. As a switched backplane has higher

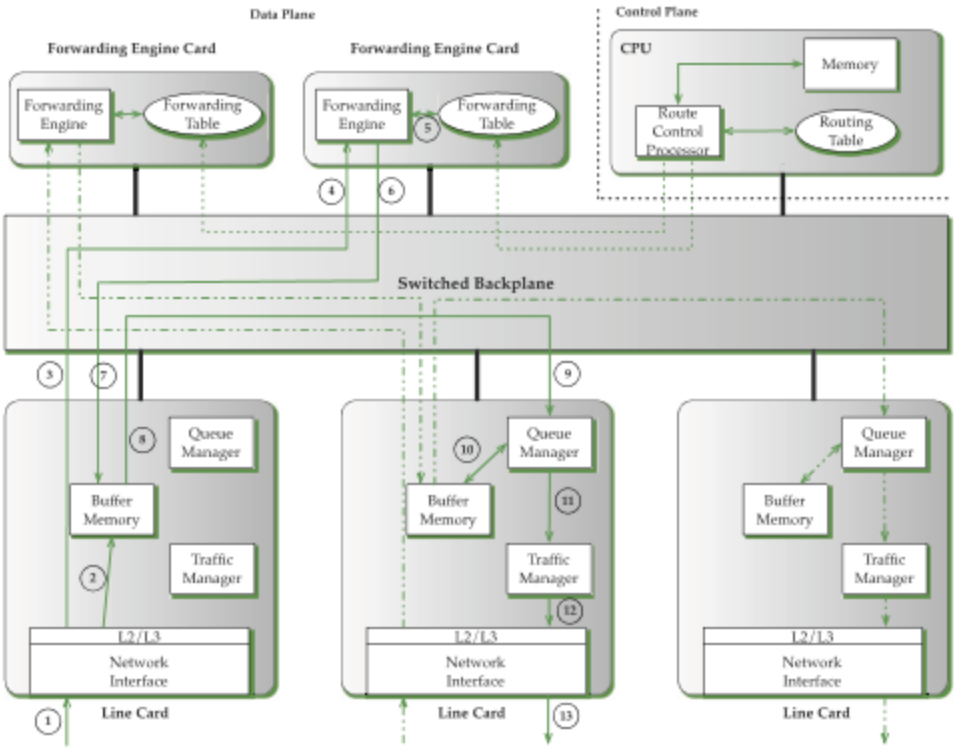


Figure 2.10: Shared forwarding engine architecture using a switched backplane.

bandwidth, a separate forward backplane is not required. Instead, both the line cards and forwarding engine cards are directly connected to the switched backplane, thus providing a communication path in which each line card can reach any forwarding engine. The control processor is also attached to the switched backplane that provides a path for updating the forwarding tables in the forwarding engine cards.

2.7.3 SHARED NOTHING ARCHITECTURES

With increasing link speeds, the architectures described so far are stretched to their limit. First, in the shared forwarding engine architecture, forwarding a packet requires traversing the backplane twice, irrespective of using two shared backplanes or a single switched backplane as shown in Figures 2.10 and 2.11. This reduces the available backplane bandwidth for forwarding packets. Second, the use of general purpose processors in the forwarding engine cards further limits the number of packets that can be processed.

A closer look at the shared forwarding engine architecture indicates that the extra hop through the backplane can be eliminated if the forwarding engine is incorporated into the line card. As routers are dedicated systems not running any specific application tasks, off-loading processing to line cards can increase the overall router performance. Further, more processing power can be added by implementing each functional module in hardware such as high-speed FPGA (field programmable gate arrays) or ASICs (application specific integrated circuits). To achieve high performance, these hardware components are interconnected by high speed links embedded in the line card.

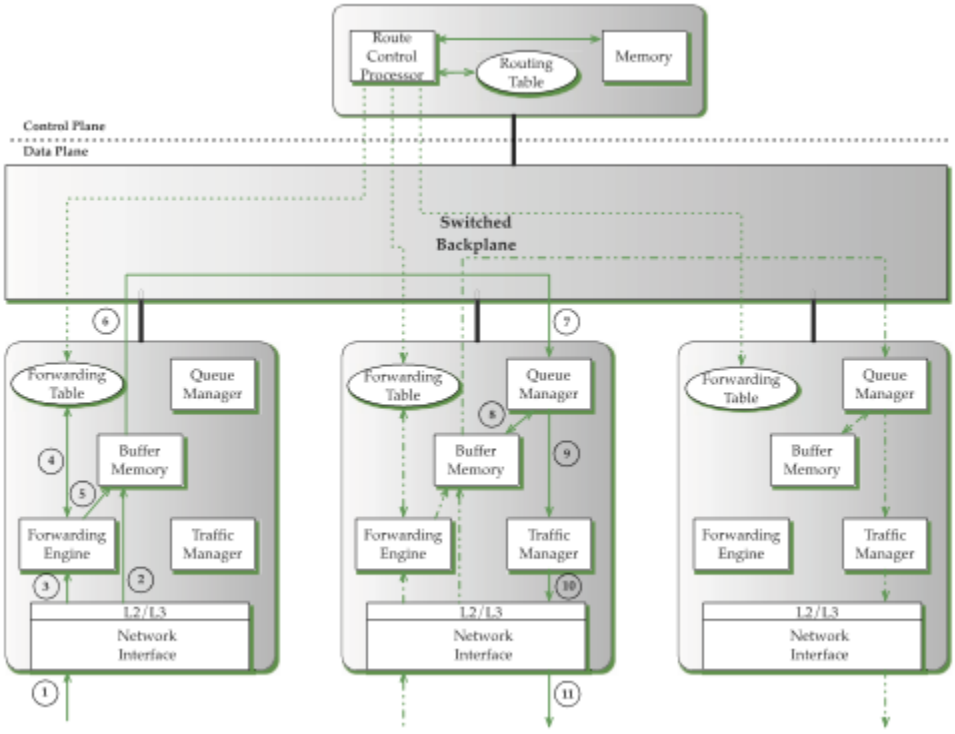


Figure 2.11: Shared nothing architecture.

A shared nothing router architecture offloads all the packet forwarding functions to the line cards.

The line cards implement these functions using custom hardware for high performance and do not share any of these components with other line cards. Hence, this architecture is named as shared nothing. Now, since the line cards are capable of handling a large number of packets, the backplane should be fast enough to handle

aggregate input from all the line cards. Hence, this architecture employs switched backplanes, which makes this setup capable of multiple transfers simultaneously. An instance of this architecture is illustrated in Figure 2.11, which also depicts the packet flow.

As you can see from Figure 2.11, all the line cards are connected to a high speed switched backplane. A packet enters the router through the network interface in the line card. It is subjected to L2/L3 processing logic, which peels off the L2 header and creates a packet context. The L2/L3 processing logic appends the L2 information such as source MAC and destination MAC to the packet context. In addition, the packet context is appended with the IP header of the packet. The L2/L3 processing module deposits the packet payload to the buffer memory and in parallel, sends the packet context along with the header to the forwarding engine.

The forwarding engine consults the forwarding table for route lookup and determines the outgoing port and egress line card. In addition to route lookup, the forwarding engine classifies the packet into a service class based on the contents of the packet header. This service class information is stored in the packet context. Depending on the result of the route lookup, the packet is extracted from the buffer memory and transmitted to the egress line card through the switched backplane. The packet is received by the queue manager in the egress line card, which stores it in the buffer memory. Depending on the priority of the packet, it is scheduled for transmission to the traffic manager. The traffic manager, based on the agreed rate of transmission, might delay the packet further or transmit it immediately on the appropriate egress interface.

The route control processor is implemented on a separate card using a general-purpose processor. In some of the architectures, this card is attached to the switched backplane for communicating to the line cards. In other architectures, a separate path for communicating to the line cards is implemented.

The processor runs routing protocols and when a route update occurs, the forwarding table is computed and propagated to the forwarding tables maintained in the line cards.

2.7.4 CLUSTERED ARCHITECTURES

One of the major limitations of routers using shared nothing architecture is the number of line cards that can be supported in a single chassis. This is because of two reasons. First, such routers are used in the core and at higher layers of aggregation where the number of links required is small but the bandwidth per link increases. Second, the packaging density possible within the racks used in central offices is limited to 19 inches (NEBS standards). In addition, a spacing of 1 inch is needed between line cards for air flow that limits the number of line cards to 16, assuming the line cards are being arranged vertically.

With the advent of dense wave-division multiplexing (DWDM) technology, each fiber can now contain many independent channels. The data rate on each channel can be as high as OC-48 (2.4 Gbps).

These channels are separated and terminated by the router with one port per channel. Hence, support for large number of ports is required. With each line card carrying only a fixed number of ports, a router needs to support a large number of line cards.

For increasing the number of line cards and the aggregate system throughput, major vendors use a clustering approach. This approach uses multiple chassis containing line cards connected to the central switch core, as shown in Figure 2.12. A variation of this approach is the use of multiple independent routers connected to a central switch core but function as a single router. In these architectures, the chassis containing line cards are connected to the switch core using very high speed optical links.

A packet entering a network interface in a line card, depending on the result of route lookup, can be destined to a line card in the same chassis or a line card in a different chassis. In the latter case, the packet must be forwarded through the switch core that sends it to the correct chassis. Once the packet reaches the chassis, it is forwarded through the appropriate egress line card.

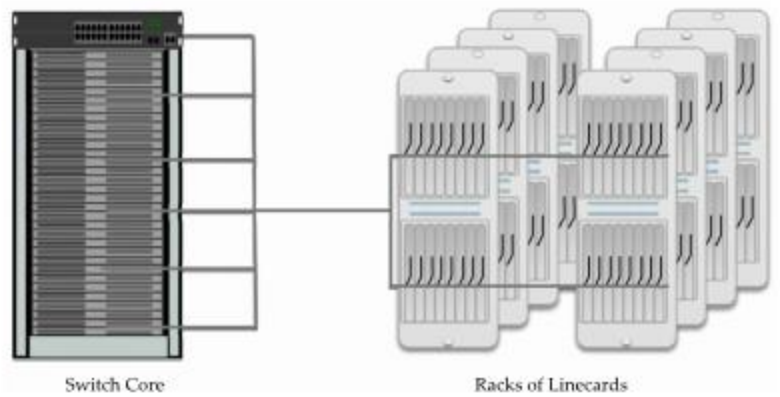


Figure 2.12: Clustered router architecture with a central switch core.

The advantage of this architecture is the ability to incrementally add the line card chassis depending on the need. A disadvantage of this architecture is that the switch core is a single point of failure. Hence, for high-availability routers, a second switch core is needed that increases the cost.

2.8 Impact of Addressing on lookup