

CHAPTER 1

Introduction

The Internet is comprised of a mesh of routers interconnected by links. Communication among nodes on the Internet (routers and end-hosts) takes place using the Internet Protocol, commonly known as IP. IP datagrams (packets) travel over links from one router to the next on their way towards their final destination. Each router performs a forwarding decision on incoming packets to determine the packet's next-hop router.

The capability to forward packets is a requirement for every IP router [3]. Additionally, an IP router may also choose to perform special processing on incoming packets. Examples of special processing include filtering packets for security reasons, delivering packets according to a pre-agreed delay guarantee, treating high priority packets preferentially, and maintaining statistics on the number of packets sent by different networks. Such special processing requires that the router classify incoming packets into one of several *flows* — all packets of a flow obey a pre-defined rule and are processed in a similar manner by the router. For example, all packets with the same source IP address may be defined to form a flow. A flow could also be defined by specific values of the destination IP address and by specific protocol values. Throughout this thesis, we will refer to routers

that classify packets into flows as *flow-aware* routers. On the other hand, flow-unaware routers treat each incoming packet individually and we will refer to them as *packet-by-packet* routers.

This thesis is about two types of algorithms: (1) algorithms that an IP router uses to decide *where* to forward packets next, and, (2) algorithms that a flow-aware router uses to *classify* packets into flows.¹ In particular, this thesis is about fast and efficient algorithms that enable routers to process many packets per second, and hence increase the capacity of the Internet.

This introductory chapter first describes the packet-by-packet router and the method it uses to make the forwarding decision, and then moves on to describe the flow-aware router and the method it uses to classify incoming packets into flows. Finally, the chapter presents the goals and metrics for evaluation of the algorithms presented later in this thesis.

1 Packet-by-packet IP router and route lookups

A packet-by-packet IP router is a special-purpose packet-switch that satisfies the requirements outlined in RFC 1812 [3] published by the Internet Engineering Task Force (IETF).² All packet-switches — by definition — perform two basic functions. First, a packet-switch must perform a forwarding decision on each arriving packet for deciding where to send it next. An IP router does this by looking up the packet's destination address in a forwarding table. This yields the address of the next-hop router³ and determines the

1. As explained later in this chapter, the algorithms in this thesis are meant for the router data-plane (i.e., the datapath of the packet), rather than the router control-plane which configures and populates the forwarding table.

2. IETF is a large international community of network equipment vendors, operators, engineers and researchers interested in the evolution of the Internet Architecture. It comprises of groups working on different areas such as routing, applications and security. It publishes several documents, called RFCs (Request For Comments). An RFC either overviews an introductory topic, or acts as a standards specification document.

3. A packet may be sent to multiple next-hop routers. Such packets are called multicast packets and are sent out on multiple egress ports. Unless explicitly mentioned, we will discuss lookups for unicast packets only.

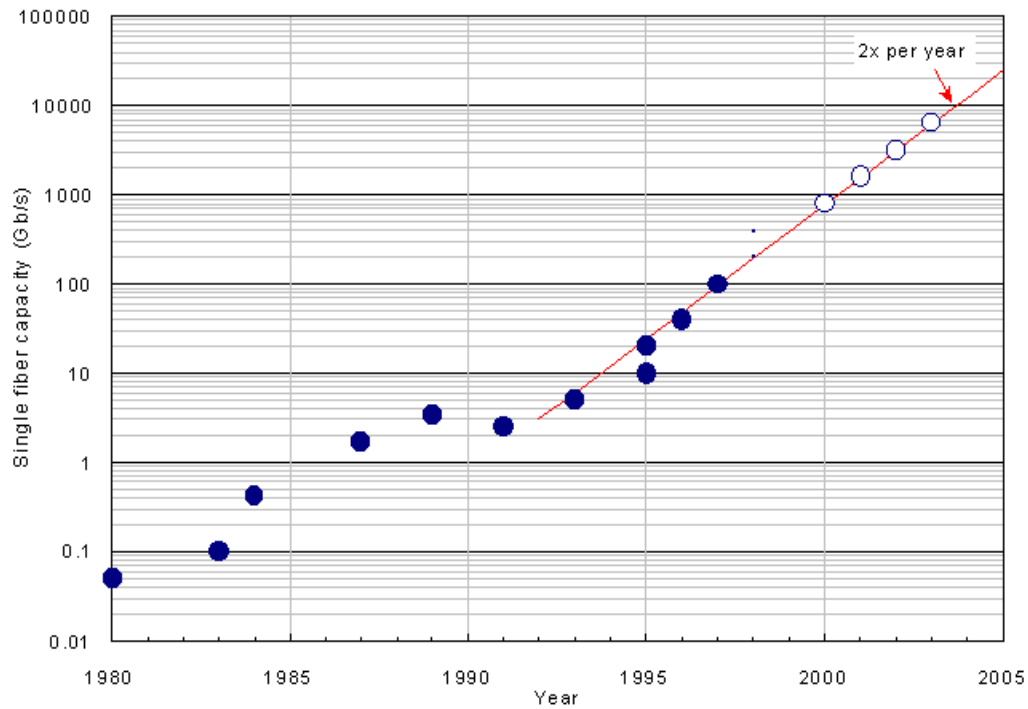


Figure 1.1 The growth in bandwidth per installed fiber between 1980 and 2005. (Source: Lucent Technologies.)

egress port through which the packet should be sent. This lookup operation is called a *route lookup* or an *address lookup* operation. Second, the packet-switch must transfer the packet from the ingress to the egress port identified by the address lookup operation. This is called *switching*, and involves physical movement of the bits carried by the packet.

The combination of route lookup and switching operations makes per-packet processing in routers a time consuming task. As a result, it has been difficult for the packet processing capacity of routers to keep up with the increased data rates of physical links in the Internet. The data rates of links have increased rapidly over the years to hundreds of gigabits per second in the year 2000 [133] — mainly because of advances in optical technologies such as WDM (Wavelength Division Multiplexing). Figure 1.1 shows the increase in bandwidth per fiber during the period 1980 to 2005, and Figure 1.2 shows the increase in

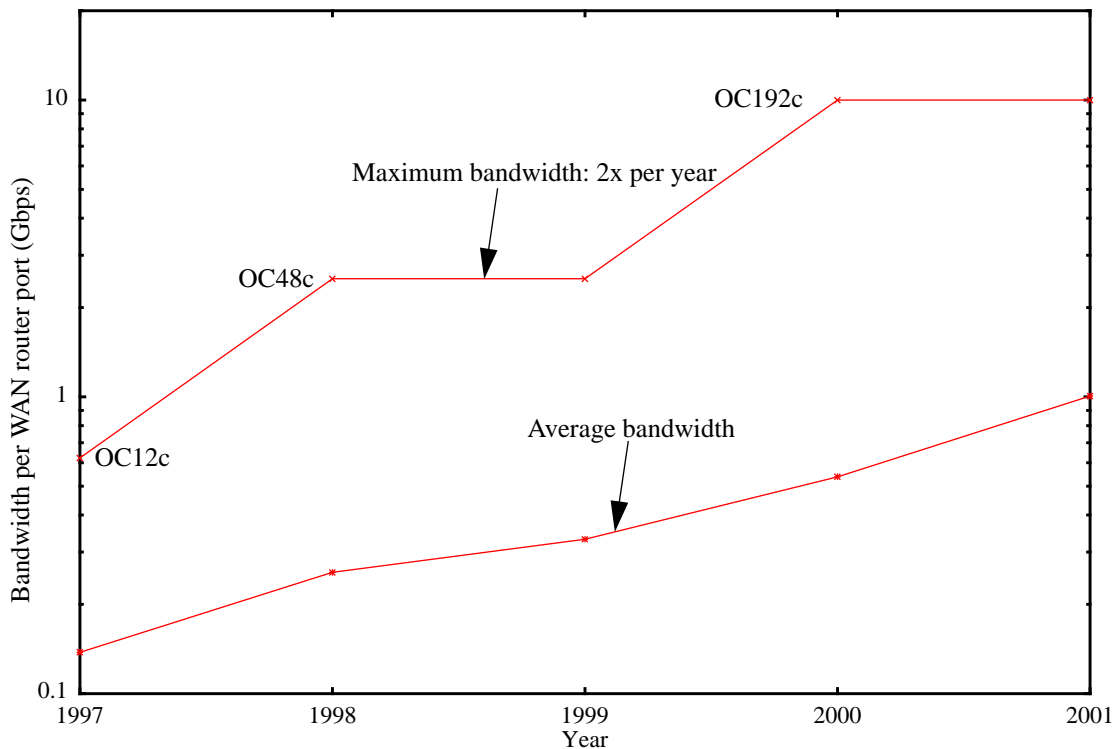


Figure 1.2 The growth in maximum bandwidth of a wide-area-network (WAN) router port between 1997 and 2001. Also shown is the average bandwidth per router port, taken over DS3, ATM OC3, ATM OC12, POS OC3, POS OC12, POS OC48, and POS OC192 ports in the WAN. (Data courtesy Dell’Oro Group, Portola Valley, CA)

the maximum bandwidth of a router port in the period 1997 to 2001. These figures highlight the gap in the data rates of routers and links — for example, in the year 2000, a data rate of 1000 Gbps is achievable per fiber, while the maximum bandwidth available is limited to 10 Gbps per router port. Figure 1.2 also shows the average bandwidth of a router port over all routers — this average is about 0.53 Gbps in the year 2000. The work presented in the first part of this thesis (Chapters 2 and 3) is motivated by the need to alleviate this mismatch in the speeds of routers and physical links — in particular, the need to perform route lookups at high speeds. High-speed switching [1][55][56][57][58][104] is an important problem in itself, but is not considered in this thesis.

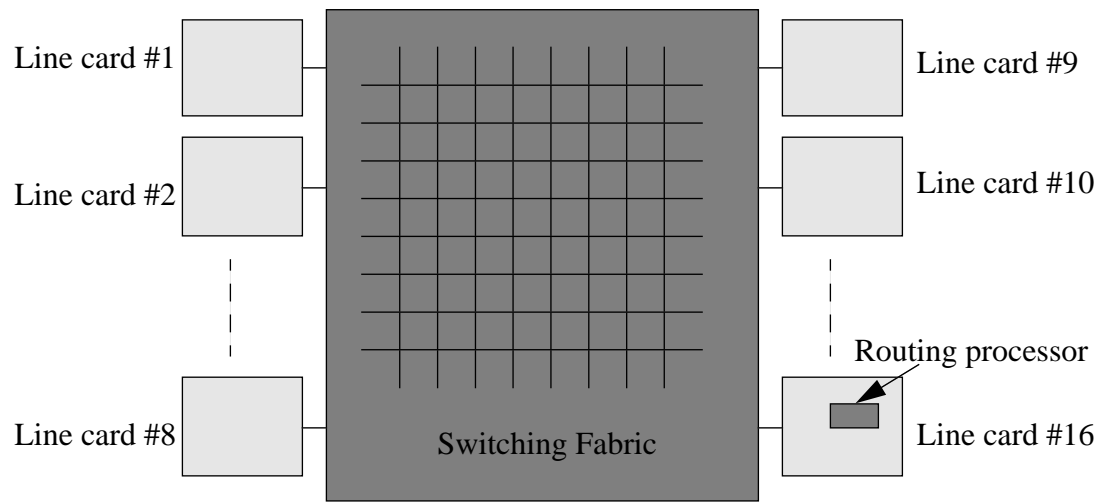


Figure 1.3 The architecture of a typical high-speed router.

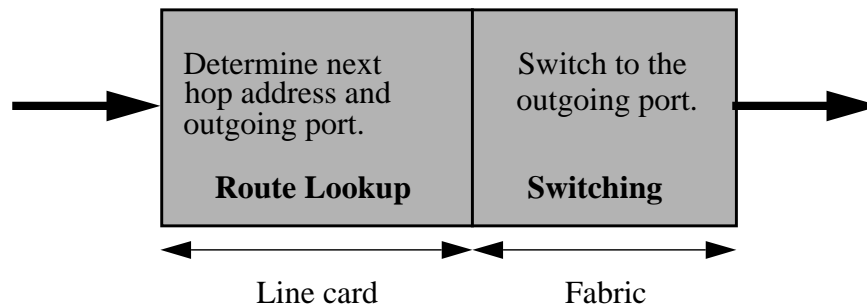


Figure 1.4 Datapath of a packet through a packet-by-packet router.

1.1 Architecture of a packet-by-packet router

Figure 1.3 shows a block diagram of the architecture of a typical high speed router. It consists of one line card for each port and a switching fabric (such as a crossbar) that interconnects all the line cards. Typically, one of the line cards houses a processor functioning as the central controller for the router. The path taken by a packet through a packet-by-packet router is shown in Figure 1.4 and consists of two main functions on the packet: (1) performing route lookup based on the packet's destination address to identify the outgoing port, and (2) switching the packet to the output port.

The routing processor in a router performs one or more routing protocols such as RIP [33][51], OSPF [65] or BGP [80] by exchanging protocol messages with neighboring routers. This enables it to maintain a *routing table* that contains a representation of the network topology state information and stores the current information about the best known paths to destination networks. The router typically maintains a version of this routing table in all line cards so that lookups on incoming packets can be performed locally on each line card, without loading the central processor. This version of the central processor's routing table is what we have been referring to as the line card's *forwarding table* because it is directly used for packet forwarding. There is another difference between the routing table in the processor and the forwarding tables in the line cards. The processor's routing table usually keeps a lot more information than the forwarding tables. For example, the forwarding table may only keep the outgoing port number, address of next-hop, and (optionally) some statistics with each route, whereas the routing table may keep additional information: e.g., time-out values, the actual paths associated with the route, etc.

The routing table is dynamic — as links go down and come back up in various parts of the Internet, routing protocol messages may cause the table to change continuously. Changes include addition and deletion of prefixes, and the modification of next-hop information for existing prefixes. The processor communicates these changes to the line card to maintain up-to-date information in the forwarding table. The need to support routing table updates has implications for the design of lookup algorithms, as we shall see later in this thesis.

1.2 Background and definition of the route lookup problem

This section explains the background of the route lookup operation by briefly describing the evolution of the Internet addressing architecture, and the manner in which this impacts the complexity of the lookup mechanism. This leads us to the formal definition of

the lookup problem, and forms a background to the lookup algorithms presented thereafter.

1.2.1 Internet addressing architecture and route lookups

In 1993, the Internet addressing architecture changed from *class-based addressing* to today's *classless addressing* architecture. This change resulted in an increase in the complexity of the route lookup operation. We first briefly describe the structure of IP addresses and the route lookup mechanism in the original class-based addressing architecture. We then describe the reasons for the adoption of classless addressing and the details of the lookup mechanism as performed by Internet routers.

IP version 4 (abbreviated as IPv4) is the version of Internet Protocol most widely used in the Internet today. IPv4 addresses are 32 bits long and are commonly written in the dotted-decimal notation — for example, 240.2.3.1, with dots separating the four bytes of the address written as decimal numbers. It is sometimes useful to view IP addresses as 32-bit unsigned numbers on the number line, $[0 \dots (2^{32} - 1)]$, which we will refer to as the **IP number line**. For example, the IP address 240.2.3.1 represents the decimal number $4026663681 \left(240 \times 2^{24} + 2 \times 2^{16} + 3 \times 2^8 + 1 \right)$ and the IP address 240.2.3.10 represents the decimal number 4026663690. Conceptually, each IPv4 address is a pair (*netid*, *hostid*), where *netid* identifies a network, and *hostid* identifies a host on that network. All hosts on the same network have the same *netid* but different *hostids*. Equivalently, the IP addresses of all hosts on the same network lie in a contiguous range on the IP number line.

The class-based Internet addressing architecture partitioned the IP address space into five classes — classes A, B and C for unicast traffic, class D for multicast traffic and class E reserved for future use. Classes were distinguished by the number of bits used to represent the *netid*. For example, a class A network consisted of a 7-bit *netid* and a 24-bit *hostid*, whereas a class C network consisted of a 21-bit *netid* and an 8-bit *hostid*. The first few

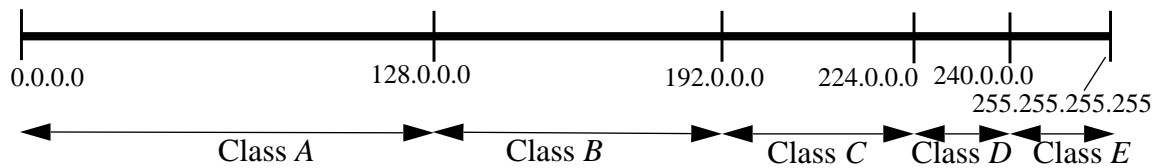


Figure 1.5 The IP number line and the original class-based addressing scheme. (The intervals represented by the classes are not drawn to scale.)

most-significant bits of an IP address determined its class, as shown in Table 1.1 and depicted on the IP number line in Figure 1.5.

TABLE 1.1. Class-based addressing.

Class	Range	Most significant address bits	netid	hostid
A	0.0.0.0 - 127.255.255.255	0	bits 1-7	bits 8-31
B	128.0.0.0 - 191.255.255.255	10	bits 2-15	bits 16-31
C	192.0.0.0 - 223.255.255.255	110	bits 3-23	bits 24-31
D (multicast)	224.0.0.0 - 239.255.255.255	1110	-	-
E (reserved for future use)	240.0.0.0 - 255.255.255.255	11110	-	-

The class-based addressing architecture enabled routers to use a relatively simple lookup operation. Typically, the forwarding table had three parts, one for each of the three unicast classes *A*, *B* and *C*. Entries in the forwarding table were tuples of the form $\langle netid, address\ of\ next\ hop \rangle$. All entries in the same part had *netids* of fixed-width — 7, 14 and 21 bits respectively for classes *A*, *B* and *C*, and the lookup operation for each incoming packet proceeded as in Figure 1.6. First, the class was determined from the most-significant bits of the packet's destination address. This in turn determined which of the three

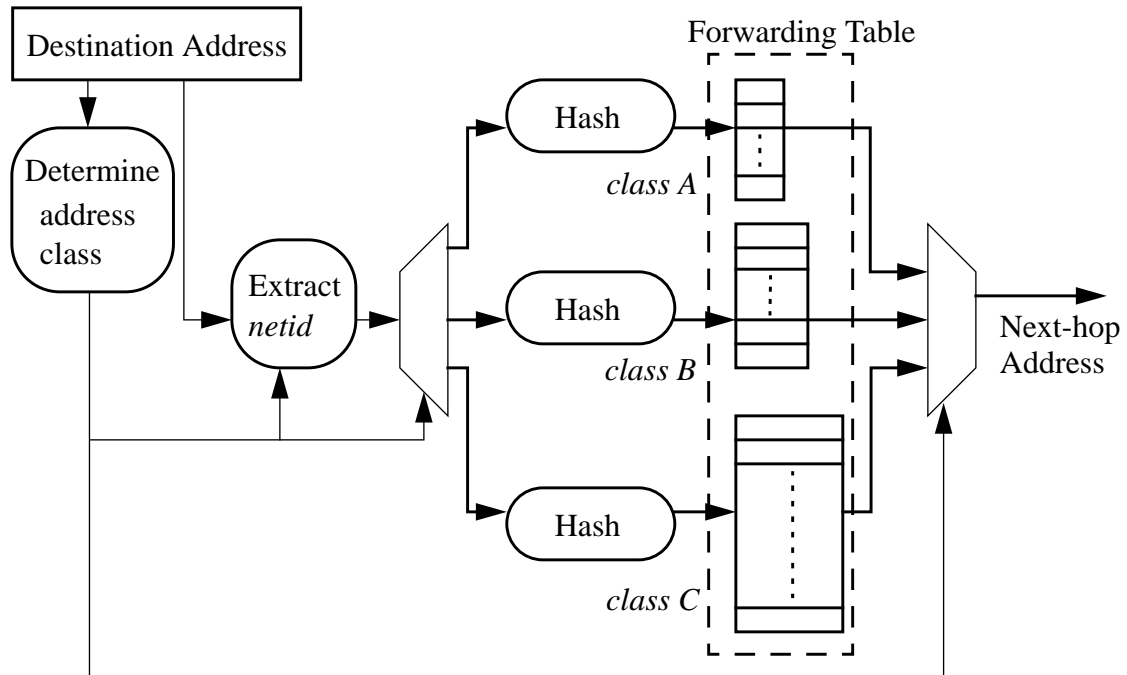


Figure 1.6 Typical implementation of the lookup operation in a class-based addressing scheme.

parts of the forwarding table to use. The router then searched for an exact match between the *netid* of the incoming packet and an entry in the selected part of the forwarding table. This exact match search could be performed using, for example, a hashing or a binary search algorithm [13].

The class-based addressing scheme worked well in the early days of the Internet. However, as the Internet grew, two problems emerged — a depletion of the IP address space, and an exponential growth of routing tables.

The allocation of network addresses on fixed *netid-hostid* boundaries (i.e., at the 8th, 16th and 24th bit positions, as shown in Table 1.1) was too inflexible, leading to a large number of wasted addresses. For example, a class *B netid* (good for 2^{16} *hostids*) had to be allocated to any organization with more than 254 hosts.¹ In 1991, it was predicted

1. While one class *C netid* accommodates 256 *hostids*, the values 0 and 255 are reserved to denote network and broadcast addresses respectively.

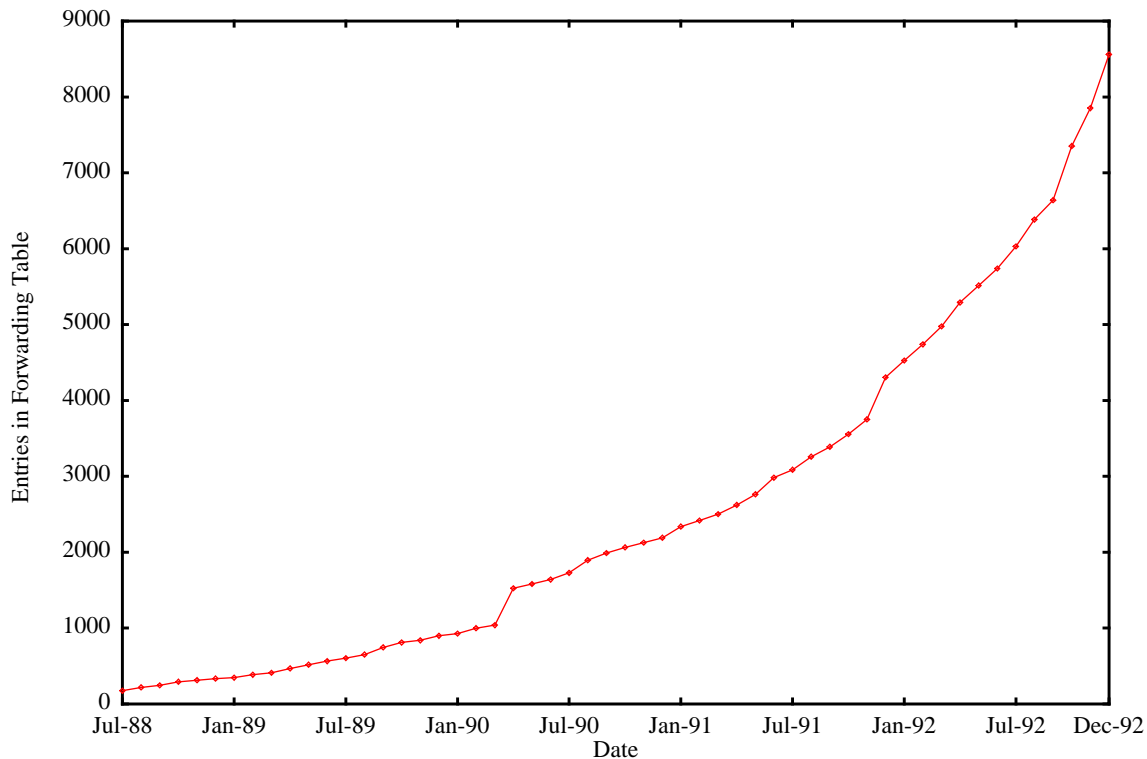


Figure 1.7 Forwarding tables in backbone routers were growing exponentially between 1988 and 1992 (i.e., under the class-based addressing scheme). (Source: RFC1519 [26])

[44][91][92] that the class B address space would be depleted in less than 14 months, and the whole IP address space would be exhausted by 1996 — even though less than 1% of the addresses allocated were actually in use [44].

The second problem was due to the fact that a backbone IP router stored every allocated *netid* in its routing table. As a result, routing tables were growing exponentially, as shown in Figure 1.7. This placed a high load on the processor and memory resources of routers in the backbone of the Internet.

In an attempt to slow down the growth of backbone routing tables and allow more efficient use of the IP address space, an alternative addressing and routing scheme called CIDR (Classless Inter-domain Routing) was adopted in 1993 [26][81]. CIDR does away

with the class-based partitioning of the IP address space and allows *netids* to be of arbitrary length rather than constraining them to be 7, 14 or 21 bits long. CIDR represents a *netid* using an IP *prefix* — a prefix of an IP address with a variable length of 0 to 32 significant bits and remaining wildcard bits.¹ An IP prefix is denoted by P/l where P is the prefix or *netid*, and l its length. For example, 192.0.1.0/24 is a 24-bit prefix that earlier belonged to class *C*. With CIDR, an organization with, say, 300 hosts can be allocated a prefix of length 23 (good for $2^{32-23} = 2^9 = 512$ *hostids*) leading to more efficient address allocation.

This adoption of variable-length prefixes now enables a hierarchical allocation of IP addresses according to the physical topology of the Internet. A service provider that connects to the Internet backbone is allocated a short prefix. The provider then allocates longer prefixes out of its own address space to other smaller Internet Service Providers (ISPs) or sites that connect to it, and so on. Hierarchical allocation allows the provider to aggregate the routing information of the sites that connect to it, before advertising routes to the routers higher up in the hierarchy. This is illustrated in the following example:

Example 1.1: (see Figure 1.8) Consider an ISP P and two sites S and T connected to P . For instance, sites S and T may be two university campuses using P 's network infrastructure for communication with the rest of the Internet. P may itself be connected to some backbone provider. Assume that P has been allocated a prefix 192.2.0.0/22, and it chooses to allocate the prefix 192.2.1.0/24 to S and 192.2.2.0/24 to T . This implies that routers in the backbone (such as $R1$ in Figure 1.8) only need to keep one table entry for the prefix 192.2.0.0/22 with P 's network as the next-hop, i.e., they do not need to keep separate routing information for individual sites S and T . Similarly, Routers inside P 's network (e.g., $R5$ and $R6$) keep entries to distinguish traffic among S and T , but not for any networks or sites that are connected downstream to S or T .

1. In practice, the shortest prefix is 8 bits long.

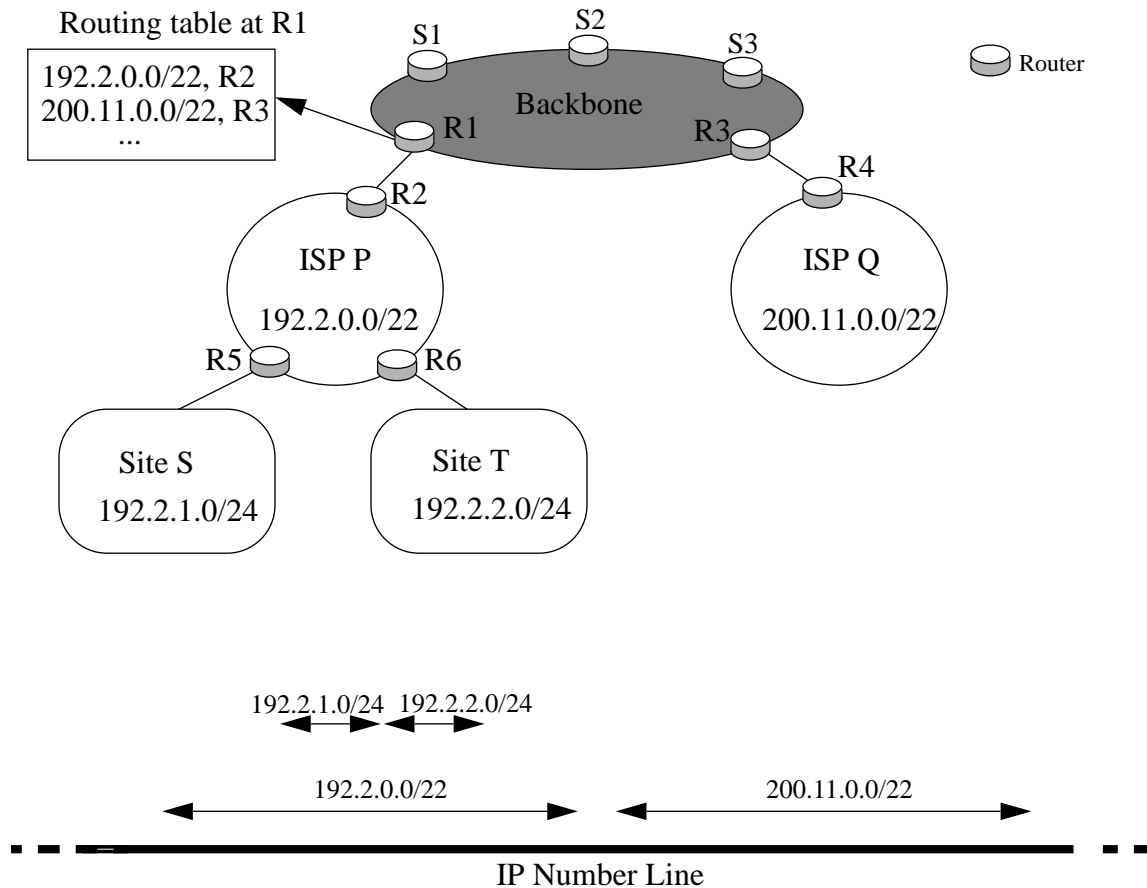


Figure 1.8 Showing how allocation of addresses consistent with the topology of the Internet helps keep the routing table size small. The prefixes are shown on the IP number line for clarity.

The aggregation of prefixes, or “route aggregation,” leads to a reduction in the size of backbone routing tables. While Figure 1.7 showed an exponential growth in the size of routing tables before widespread adoption of CIDR in 1994, Figure 1.9 shows that the growth turned linear thereafter — at least till January 1998, since when it seems to have become faster than linear again.¹

1. It is a bit premature to assert that routing tables are again growing exponentially. In fact, the portion of the plot in Figure 1.9 after January 1998 fits well with an exponential as well as a quadratic curve. While not known definitively, the increased rate of growth could be because: (1) Falling costs of raw transmission bandwidth are encouraging decreased aggregation and a finer mesh of granularity; (2) Increasing expectations of reliability are forcing network operators to make their sites multi-homed.

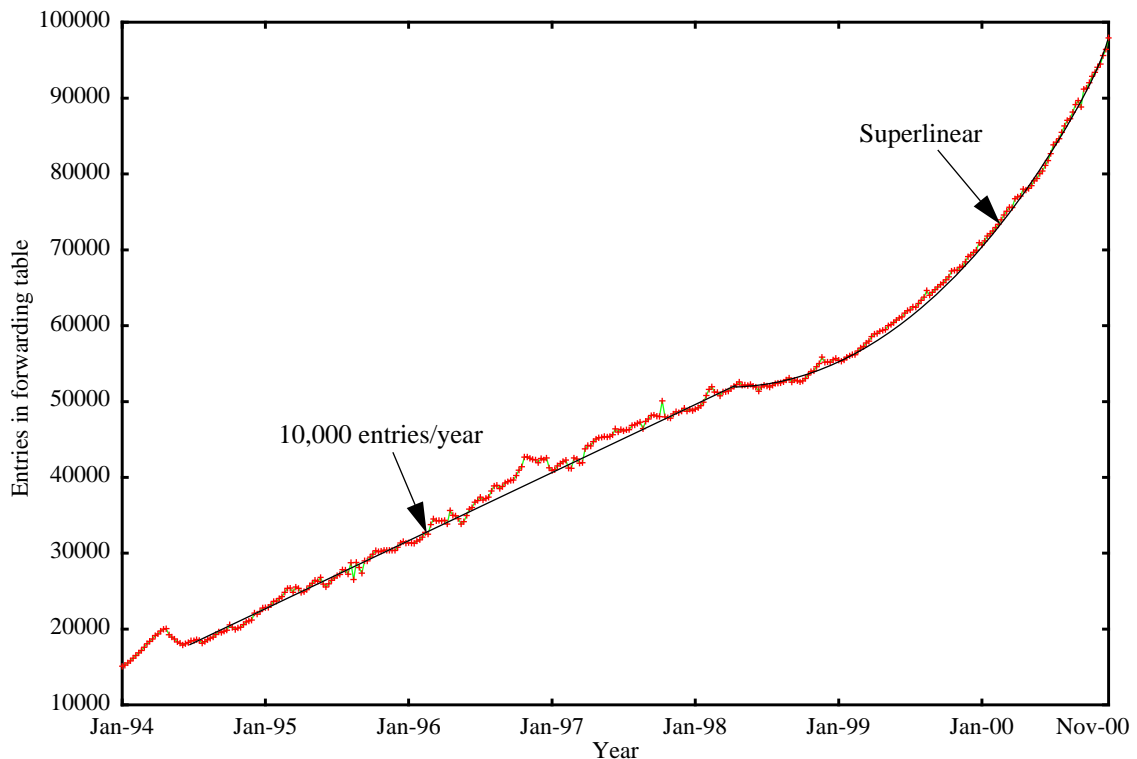


Figure 1.9 This graph shows the weekly average size of a backbone forwarding table (source [136]). The dip in early 1994 shows the immediate effect of widespread deployment of CIDR.

Hierarchical aggregation of addresses creates a new problem. When a site changes its service provider, it would prefer to keep its prefix (even though topologically, it is connected to the new provider). This creates a “hole” in the address space of the original provider — and so this provider must now create specific entries in its routing tables to allow correct forwarding of packets to the moved site. Because of the presence of specific entries, routers are required to be able to forward packets according to the *most specific route* present in their forwarding tables. The same capability is required when a site is multi-homed, i.e., has more than one connection to an upstream carrier or a backbone provider. The following examples make this clear:

Example 1.2: Assume that site T in Figure 1.8 with address space 192.2.2.0/24 changed its ISP to Q, as shown in Figure 1.10. The routing table at router R1 needs to have an additional entry corresponding to 192.2.2.0/24 pointing to Q’s network. Packets des-

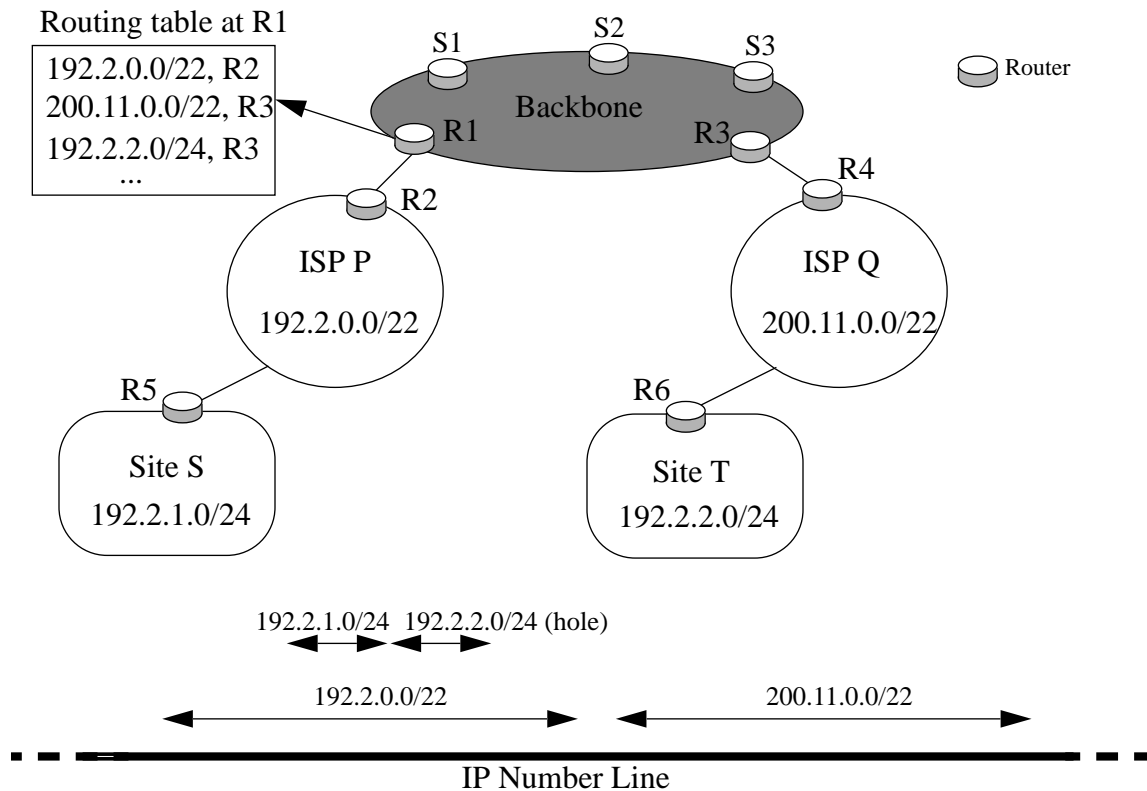


Figure 1.10 Showing the need for a routing lookup to find the most specific route in a CIDR environment.

tinged to T at router R1 match this more specific route and are correctly forwarded to the intended destination in T (see Figure 1.10).

Example 1.3: Assume that ISP Q of Figure 1.8 is multi-homed, being connected to the backbone also through routers S4 and R7 (see Figure 1.11). The portion of Q's network identified with the prefix 200.11.1.0/24 is now better reached through router R7. Hence, the forwarding tables in backbone routers need to have a separate entry for this special case.

Lookups in the CIDR environment

With CIDR, a router's forwarding table consists of entries of the form $\langle route\text{-}prefix, next\text{-}hop\text{-}addr \rangle$, where *route-prefix* is an IP prefix and *next-hop-addr* is the IP address of the next hop. A destination address *matches* a route-prefix if the significant bits of the pre-

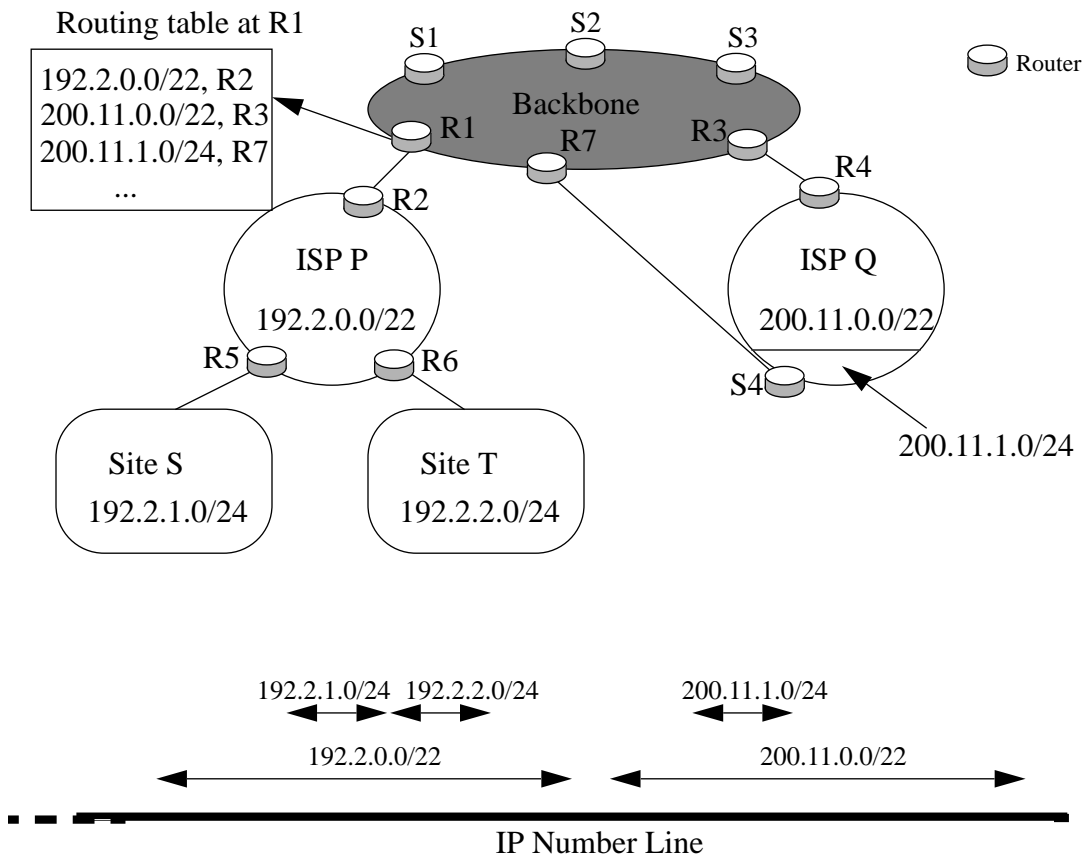


Figure 1.11 Showing how multi-homing creates special cases and hinders aggregation of prefixes.

fix are identical to the first few bits of the destination address. A routing lookup operation on an incoming packet requires the router to find the most specific route for the packet. This implies that the router needs to solve the **longest prefix matching problem**, defined as follows.

Definition 1.1: *The **longest prefix matching** problem is the problem of finding the forwarding table entry containing the longest prefix among all prefixes (in other forwarding table entries) matching the incoming packet's destination address. This longest prefix is called the longest matching prefix.*

Example 1.4: The forwarding table in router R1 of Figure 1.10 is shown in Table 1.2. If an incoming packet at this router has a destination address of 200.11.0.1, it will match only the prefix 200.11.0.0/22 (entry #2) and hence will be forwarded to router R3.

If the packet's destination address is 192.2.2.4, it matches two prefixes (entries #1 and #3). Because entry #3 has the longest matching prefix, the packet will be forwarded to router R3.

TABLE 1.2. The forwarding table of router R1 in Figure 1.10.

Entry Number	Prefix	Next-Hop
1.	192.2.0.0/22	R2
2.	200.11.0.0/22	R3
3.	192.2.2.0/24	R3

Difficulty of longest prefix matching

The destination address of an arriving packet does not carry with it the information needed to determine the length of the longest matching prefix. Hence, we cannot find the longest match using an exact match search algorithm (for example, using hashing or a binary search procedure). Instead, a search for the longest matching prefix needs to determine both the length of the longest matching prefix as well as the forwarding table entry containing the prefix of this length that matches the incoming packet's destination address. One naive longest prefix matching algorithm is to perform 32 different exact match search operations, one each for all prefixes of length i , $1 \leq i \leq 32$. This algorithm would require 32 exact match search operations. As we will see later in this thesis, faster algorithms are possible.

In summary, the need to perform longest prefix matching has made routing lookups more complicated now than they were before the adoption of CIDR when only one exact match search operation was required. Chapters 2 and 3 of this thesis will present efficient longest prefix matching algorithms for fast routing lookups.

2 Flow-aware IP router and packet classification

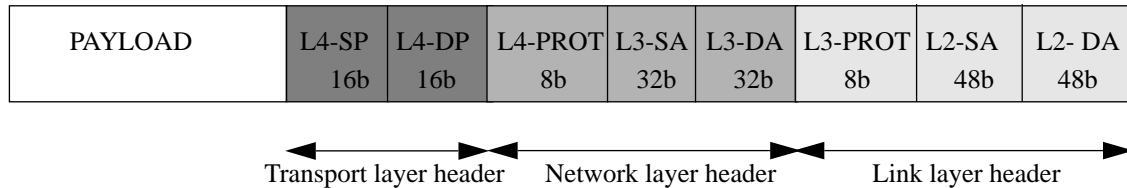
As mentioned earlier, routers may optionally classify packets into flows for special processing. In this section, we first describe why some routers are flow-aware, and how they use packet classification to recognize flows. We also provide a brief overview of the architecture of flow-aware routers. We then provide the background leading to the formal definition of the packet classification problem. Fast packet classification is the subject of the second part of this thesis (Chapters 4 and 5).

2.1 Motivation

One main reason for the existence of flow-aware routers stems from an ISP's desire to have the capability of providing differentiated services to its users. Traditionally, the Internet provides only a "best-effort" service, treating all packets going to the same destination identically, and servicing them in a first-come-first-served manner. However, the rapid growth of the Internet has caused increasing congestion and packet loss at intermediate routers. As a result, some users are willing to pay a premium price in return for better service from the network. To maximize their revenue, the ISPs also wish to provide different levels of service at different prices to users based on their requirements, while still deploying one common network infrastructure.¹

In order to provide differentiated services, routers require additional mechanisms. These mechanisms — admission control, conditioning (metering, marking, shaping, and policing), resource reservation (optional), queue management and fair scheduling (such as weighted fair queueing) — require, first of all, the capability to distinguish and isolate traffic belonging to different users based on service agreements negotiated between the ISP and its customer. This has led to demand for flow-aware routers that negotiate these

1. This is analogous to the airlines, who also provide differentiated services (such as economy and business class) to different users based on their requirements, while still using the same common infrastructure.



L2 = Layer 2 (e.g., Ethernet)

L3 = Layer 3 (e.g., IP)

L4 = Layer 4 (e.g., TCP)

DA = Destination Address

SA = Source Address

PROT = Protocol

SP = Source Port

DP = Destination Port

Figure 1.12 This figure shows some of the header fields (and their widths) that might be used for classifying a packet. Although not shown in this figure, higher layer (e.g., application-layer) fields may also be used for packet classification.

service agreements, express them in terms of *rules* or *policies* configured on incoming packets, and isolate incoming traffic according to these rules.

We call a collection of rules or policies a *policy database*, *flow classifier*, or simply a *classifier*.¹ Each rule specifies a flow that a packet may belong to based on some criteria on the contents of the packet header, as shown in Figure 1.12. All packets belonging to the same flow are treated in a similar manner. The identified flow of an incoming packet specifies an *action* to be applied to the packet. For example, a firewall router may carry out the action of either *denying* or *allowing* access to a protected network. The determination of this action is called *packet classification* — the capability of routers to identify the action associated with the “best” rule an incoming packet matches. Packet classification allows ISPs to differentiate from their competition and gain additional revenue by providing different value-added services to different customers.

1. Sometimes, the functional datapath element that classifies packets is referred to as a classifier. In this thesis, however, we will consistently refer to the policy database as a classifier.

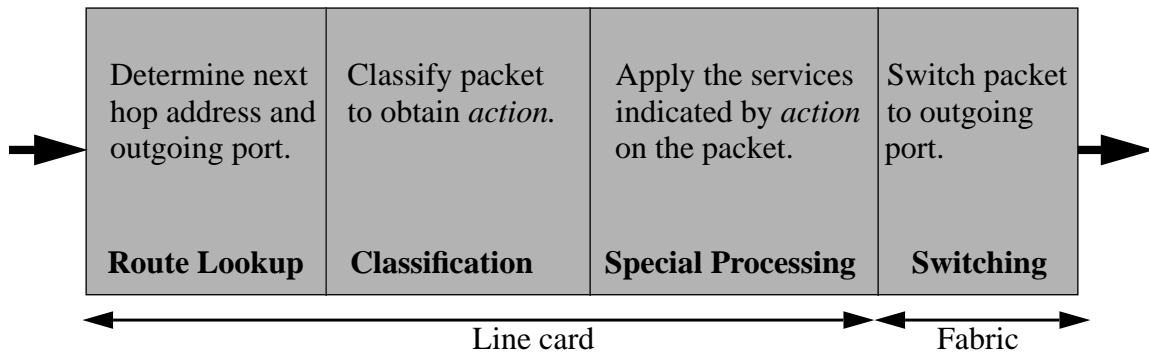


Figure 1.13 Datapath of a packet through a flow-aware router. Note that in some applications, a packet may need to be classified both before and after route lookup.

2.2 Architecture of a flow-aware router

Flow-aware routers perform a superset of the functions of a packet-by-packet router. The typical path taken by a packet through a flow-aware router is shown in Figure 1.13 and consists of four main functions on the packet: (1) performing route lookup to identify the outgoing port, (2) performing classification to identify the flow to which an incoming packet belongs, (3) applying the action (as part of the provisioning of differentiated services or some other form of special processing) based on the result of classification, and (4) switching to the output port. The various forms of special processing in function (3), while interesting in their own right, are not the subject of this thesis. The following references describe a variety of actions that a router may perform: admission control [42], queueing [25], resource reservation [6], output link scheduling [18][74][75][89] and billing [21].

2.3 Background and definition of the packet classification problem

Packet classification enables a number of additional, non-best-effort network services other than the provisioning of differentiated qualities of service. One of the well-known applications of packet classification is a firewall. Other network services that require packet classification include policy-based routing, traffic rate-limiting and policing, traffic

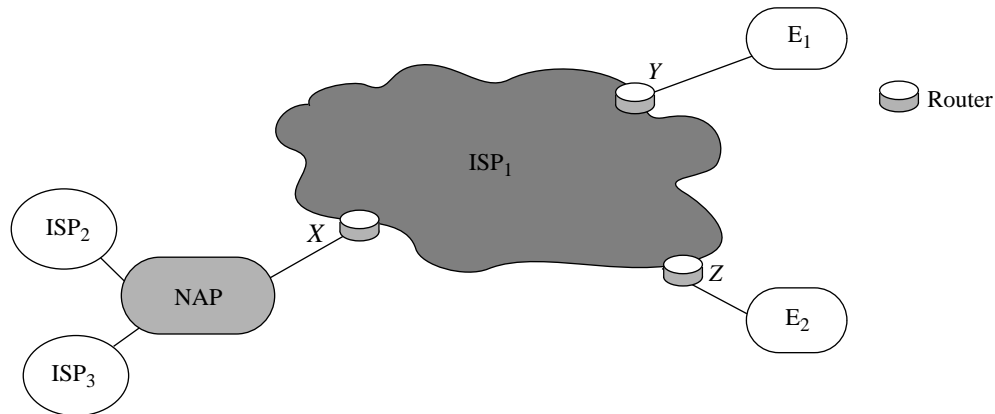


Figure 1.14 Example network of an ISP (ISP₁) connected to two enterprise networks (E₁ and E₂) and to two other ISP networks across a network access point (NAP).

shaping, and billing. In each case, it is necessary to determine which flow an arriving packet belongs to so as to determine — for example — whether to forward or filter it, where to forward it to, what type of service it should receive, or how much should be charged for transporting it.

TABLE 1.3. Some examples of value-added services.

Service	Example
Packet Filtering	Deny all traffic from ISP ₃ (on interface X) destined to E ₂ .
Policy Routing	Send all voice-over-IP traffic arriving from E ₁ (on interface Y) and destined to E ₂ via a separate ATM network.
Accounting & Billing	Treat all video traffic to E ₁ (via interface Y) as highest priority and perform accounting for such traffic.
Traffic Rate-limiting	Ensure that ISP ₂ does not inject more than 10 Mbps of email traffic and 50 Mbps of total traffic on interface X.
Traffic Shaping	Ensure that no more than 50 Mbps of web traffic is sent to ISP ₂ on interface X.

To help illustrate the variety of packet classifiers, let us consider some examples of how packet classification can be used by an ISP to provide different services. Figure 1.14 shows ISP₁ connected to three different sites: two enterprise networks E₁ and E₂, and a

Network Access Point¹ (NAP), which is in turn connected to two other ISPs — ISP₂ and ISP₃. ISP₁ provides a number of different services to its customers, as shown in Table 1.3.

Table 1.4 shows the categories that an incoming packet must be classified into by the router at interface X . Note that the classes specified may or may not be mutually exclusive.

TABLE 1.4. Given the rules in Table 1.3, the router at interface X must classify an incoming packet into the following categories.

Service	Flow	Relevant Packet Fields
Packet Filtering	From ISP ₃ and going to E ₂	Source link-layer address, destination network-layer address
Traffic rate-limiting	Email and from ISP ₂	Source link-layer address, source transport port number
Traffic shaping	Web and to ISP ₂	Destination link-layer address, destination transport port number
	All other packets	—

For example, the first and second flow in Table 1.4 overlap. This happens commonly, and when no explicit priorities are specified, we follow the convention that rules closer to the top of the list have higher priority.

With this background, we proceed to define the problem of packet classification.

Each rule of the classifier has d components. The i^{th} component of rule R , denoted as $R[i]$, is a regular expression on the i^{th} field of the packet header. A packet P is said to *match* a particular rule R , if $\forall i$, the i^{th} field of the header of P satisfies the regular expression $R[i]$. In practice, a rule component is not a general regular expression — often limited by syntax to a simple address/mask or operator/number(s) specification. In an address/mask specification, a 0 at bit position x in the mask denotes that the corresponding bit in

1. A network access point (NAP) is a network location which acts as an exchange point for Internet traffic. An ISP connects to a NAP to exchange traffic with other ISPs at that NAP.

the address is a “don’t care” bit. Similarly, a 1 at bit position x in the mask denotes that the corresponding bit in the address is a significant bit. For instance, the first and third most significant bytes in a packet field matching the specification $171.4.3.4/255.0.255.0$ must be equal to 171 and 3, respectively, while the second and fourth bytes can have any value. Examples of operator/number(s) specifications are *eq 1232* and *range 34-9339*, which specify that the matching field value of an incoming packet must be equal to 1232 in the former specification, and can have any value between 34 and 9339 (both inclusive) in the latter specification. Note that a route-prefix can be specified as an address/mask pair where the mask is *contiguous* — i.e., all bits with value 1 appear to the left of (i.e., are more significant than) bits with value 0 in the mask. For instance, the mask for an 8-bit prefix is 255.0.0.0. A route-prefix of length l can also be specified as a range of width equal to 2^t where $t = 32 - l$. In fact, most of the commonly occurring specifications in practice can be viewed as range specifications.

We can now formally define packet classification:

Definition 1.2: A classifier C has N rules, R_j , $1 \leq j \leq N$, where R_j consists of three entities — (1) A regular expression $R_j[i]$, $1 \leq i \leq d$, on each of the d header fields, (2) A number $\text{pri}(R_j)$, indicating the priority of the rule in the classifier, and (3) An action, referred to as $\text{action}(R_j)$. For an incoming packet P with the header considered as a d -tuple of points (P_1, P_2, \dots, P_d) , the **d -dimensional packet classification** problem is to find the rule R_m with the highest priority among all rules R_j matching the d -tuple; i.e., $\text{pri}(R_m) > \text{pri}(R_j)$, $\forall j \neq m$, $1 \leq j \leq N$, such that P_i matches $R_j[i]$, $\forall (1 \leq i \leq d)$. We call rule R_m the best matching rule for packet P .

Example 1.5: An example of a classifier in four dimensions is shown in Table 1.5. By convention, the first rule R1 has the highest priority and rule R7 has the lowest priority (* denotes a complete wildcard specification, and ‘gt v’ denotes any value greater than v). Classification results on some example packets using this classifier are shown in Table 1.6.

TABLE 1.5. An example classifier.

Rule	Network-layer destination (address/mask)	Network-layer source (address/mask)	Transport-layer destination	Transport-layer protocol	Action
R1	152.163.190.69/ 255.255.255.255	152.163.80.11/ 255.255.255.255	*	*	Deny
R2	152.168.3.0/ 255.255.255.0	152.163.200.157/ 255.255.255.255	eq http	udp	Deny
R3	152.168.3.0/ 255.255.255.0	152.163.200.157/ 255.255.255.255	range 20-21	udp	Permit
R4	152.168.3.0/ 255.255.255.0	152.163.200.157/ 255.255.255.255	eq http	tcp	Deny
R5	152.163.198.4/ 255.255.255.255	152.163.161.0/ 255.255.252.0	gt 1023	tcp	Permit
R6	152.163.198.4/ 255.255.255.255	152.163.0.0/ 255.255.0.0	gt 1023	tcp	Deny
R7	*	*	*	*	Permit

TABLE 1.6. Examples of packet classification on some incoming packets using the classifier of Table 1.5.

Packet Header	Network-layer destination address	Network-layer source address	Transport-layer destination port	Transport-layer protocol	Best matching rule, action
P1	152.163.190.69	152.163.80.11	http	tcp	R1, Deny
P2	152.168.3.21	152.163.200.157	http	udp	R2, Deny
P3	152.168.198.4	152.163.160.10	1024	tcp	R5, Permit

We can see that routing lookup is an instance of one-dimensional packet classification. In this case, all packets destined to the set of addresses described by a common prefix may be considered to be part of the same flow. Each rule has a route-prefix as its only compo-

ment and has the next hop address associated with this prefix as the action. If we define the priority of the rule to be the length of the route-prefix, determining the longest-matching prefix for an incoming packet is equivalent to determining the best matching rule in the classifier. The packet classification problem is therefore a generalization of the routing lookup problem. Chapters 4 and 5 of this thesis will present efficient algorithms for fast packet classification in flow-aware routers.

3 Goals and metrics for lookup and classification algorithms

A lookup or classification algorithm preprocesses a routing table or a classifier to compute a data structure that is then used to lookup or classify incoming packets. This preprocessing is typically done in software in the routing processor, discussed in Section 1.1. There are a number of properties that we desire for all lookup and classification algorithms:

- High speed.
- Low storage requirements.
- Flexibility in implementation.
- Ability to handle large real-life routing tables and classifiers.
- Low preprocessing time.
- Low update time.
- Scalability in the number of header fields (for classification algorithms only).
- Flexibility in specification (for classification algorithms only).

We now discuss each of these properties in detail.

- *High speed* — Increasing data rates of physical links require faster address lookups at routers. For example, links running at OC192c (approximately 10 Gbps) rates need the router to process 31.25 million packets per second (assuming mini-

mum-sized 40 byte TCP/IP packets).¹ We generally require algorithms to perform well in the worst case, e.g., classify packets at wire-speed. If this were not the case, all packets (regardless of the flow they belong to) would need to be queued before the classification function. This would defeat the purpose of distinguishing and isolating flows, and applying different actions on them. For example, it would make it much harder to control the delay of a flow through the router. At the same time, in some applications, for example, those that do not provide qualities of service, a lookup or classification algorithm that performs well in the *average* case may be acceptable, in fact desirable, because the average lookup performance can be much higher than the worst-case performance. For such applications, the algorithm needs to process packets at the rate of 3.53 million packets per second for OC192c links, assuming an average Internet packet size of approximately 354 bytes [121]. Table 1.7 lists the lookup performance required in one router port to

TABLE 1.7. Lookup performance required as a function of line-rate and packet size.

Year	Line	Line-rate (Gbps)	40-byte packets (Mpps)	84-byte packets (Mpps)	354-byte packets (Mpps)
1995-7	T1	0.0015	0.0468	0.0022	0.00053
1996-8	OC3c	0.155	0.48	0.23	0.054
1997-8	OC12c	0.622	1.94	0.92	0.22
1999-2000	OC48c	2.50	7.81	3.72	0.88
(Now) 2000-1	OC192c	10.0	31.2	14.9	3.53
(Next) 2002-3	OC768c	40.0	125.0	59.5	14.1
1997-2000	1 Gigabit-Ethernet	1.0	N.A.	1.49	0.35

1. In practice, IP packets are encapsulated and framed before being sent on SONET links. The most commonly used encapsulation method is PPP (Point-to-Point Protocol) in HDLC-like framing. (HDLC stands for High-level Data Link Control). This adds either 7 or 9 bytes of overhead (1 byte flag, 1 byte address, 1 byte control, 2 bytes protocol and 2 to 4 bytes of frame check sequence fields) to the packet. When combined with the SONET overhead (27 bytes of line and section overhead in a 810 byte frame), the lookup rate required for 40 byte TCP/IP packets becomes approximately 25.6 Mpps. (Please see IETF RFC 1661/1662 for PPP/HDLC framing and RFC 1619/2615 for PPP over SONET.)

handle a continuous stream of incoming packets of a given size (84 bytes is the minimum size of a Gigabit-Ethernet frame — this includes a 64-byte packet, 7-byte preamble, 1-byte start-of-frame delimiter, and 12 bytes of inter-frame gap).

- *Flexibility in implementation* — The forwarding engine may be implemented either in software or in hardware depending upon the system requirements. Thus, a lookup or classification algorithm should be suitable for implementation in both hardware and software. For the highest speeds (e.g., for OC192c in the year 2000), we expect that hardware implementation will be necessary — hence, the algorithm design should be amenable to pipelined implementation.
- *Low storage requirements* — We desire that the storage requirements of the data structure computed by the algorithm be small. Small storage requirements enable the use of fast but expensive memory technologies like SRAMs (Synchronous Random Access Memories). A memory-efficient algorithm can benefit from an on-chip cache if implemented in software, and from an on-chip SRAM if implemented in hardware.
- *Ability to handle large real-life routing tables and classifiers* — The algorithm should scale well both in terms of storage and speed with the size of the forwarding table or the classifier. At the time of the writing of this thesis, the forwarding tables of backbone routers contain approximately 98,000 route-prefixes and are growing rapidly (as shown in Figure 1.9). A lookup engine deployed in the year 2001 should be able to support approximately 400,000-512,000 prefixes in order to be useful for at least five years. Therefore, lookup and classification algorithms should demonstrate good performance on current real-life routing tables and classifiers, as well as accommodate future growth.
- *Low preprocessing time* — Preprocessing time is the time taken by an algorithm to compute the initial data structure. An algorithm that supports incremental updates of its data structure is said to be “*dynamic*.” A “*static*” algorithm requires the whole data structure to be recomputed each time a rule is added or

deleted. In general, dynamic algorithms can tolerate larger preprocessing times than static algorithms. (The absolute values differ with applications.)

- *Low update time* — Routing tables have been found to change fairly frequently, often at the peak rate of a few hundred prefixes per second and at the average rate of more than a few prefixes per second [47]. A lookup algorithm should be able to update the data structure at least this fast. For classification algorithms, the update rate differs widely among different applications — a very low update rate may be sufficient in firewalls where entries are added manually or infrequently. On the other hand, a classification algorithm must be able to support a high update rate in so called “stateful” classifiers where a packet may dynamically trigger the addition or deletion of a new rule or a fine-granularity flow.
- *Scalability in the number of header fields* (for classification algorithms only) — A classification algorithm should ideally allow matching on arbitrary fields, including link-layer, network-layer, transport-layer and — in some cases — the application-layer headers.¹ For instance, URL (universal resource locator — the identifier used to locate resources on the World Wide Web) based classification may be used to route a user’s packets across a different network or to give the user a different quality of service. Hence, while it makes sense to optimize for the commonly used header fields, the classification algorithm should not preclude the use of other header fields.
- *Flexibility in specification* (for classification algorithms only) — A classification algorithm should support flexible rule specifications, including prefixes, operators (such as range, less than, greater than, equal to, etc.) and wildcards. Even non-contiguous masks may be required, depending on the application using classification.

1. That is why packet-classifying routers have sometimes been called “layerless switches”.

4 Outline of the thesis

This thesis proposes several novel lookup and classification algorithms. There is one chapter devoted to each algorithm. Each chapter first presents background work related to the algorithm. It then presents the motivation, key concepts, properties, and implementation results for the algorithm. It also evaluates the algorithm against the metrics outlined above and against previous work on the subject.

Chapter 2 presents an overview of previous work on routing lookups. It proposes and discusses a simple routing lookup algorithm optimized for implementation in dedicated hardware. This algorithm performs the longest prefix matching operation in two memory accesses that can be pipelined to give the throughput of one routing lookup every memory access. This corresponds to 20 million packets per second with 50 ns DRAMs (Dynamic Random Access Memories).

With the motivation of high speed routing lookups, Chapter 3 defines a new problem of minimizing the average lookup time while keeping the maximum lookup time bounded. This chapter then describes and analyzes two algorithms to solve this new problem. Experiments show an improvement by a factor of 1.7 in the average number of memory accesses per lookup over those obtained by worst-case lookup time minimization algorithms. Moreover, the algorithms proposed in this chapter support almost perfect balancing of the incoming lookup load, making them easily parallelizable for high speed designs.

In Chapter 4, we move on to the problem of multi-field packet classification. Chapter 4 provides an overview of previous work and highlights the issues in designing solutions for this problem. This chapter proposes and discusses the performance of a novel algorithm for fast classification on multiple header fields.

Chapter 5 presents another new algorithm for high speed multi-field packet classification. This algorithm is different from the one proposed in Chapter 4 in that it supports fast incremental updates, is otherwise slower, and occupies a smaller amount of storage.

Finally, Chapter 6 concludes by discussing directions for future work in the area of fast routing lookups and packet classification.

