CHAPTER

CLOUD ACCESS AND CLOUD INTERCONNECTION NETWORKS 5

The decades-long evolution of microprocessor and storage technologies, computer architecture and software systems, parallel algorithms and distributed control strategies, paved the way to cloud computing. The interconnectivity supported by a continually evolving Internet made cloud computing feasible. A cloud is built around a high-performance interconnect, the servers of a cloud infrastructure communicate through high-bandwidth and low-latency networks. Unquestionably, communication is at the heart of cloud computing.

At the scale of a single system optimal performance requires a balance between the bandwidth, the number of operations per unit of time, of the three major subsystems, CPU, memory, and I/O. The reality is that processor bandwidth is orders of magnitude higher than the I/O bandwidth and much higher than the memory bandwidth. This imbalance is further amplified by a well-known empirical law, the Moore's Law whose corollary is that the processor performance doubles every 18 months or so, much faster than memory and I/O.

The effects of this imbalance are inevitably amplified in a large-scale system where the interconnect allows a very large number of processors to work together under the control of the orchestration software. The designers of a cloud computing infrastructure are acutely aware that the communication bandwidth goes down and the communication latency increases the farther from the CPU data travels. The limits of cloud interconnection networks latency and bandwidth are tested by the demands of a cloud infrastructure with millions of servers.

Cloud workloads fall into four broad categories based on their dominant resource needs: CPU-intensive, memory-intensive, I/O-intensive, and storage-intensive. While the first two benefit from, but do not require, high-performing networking, the last two do. Networking performance directly impacts the performance of I/O- and storage-intensive workloads.

A recent report [454] forecasts that by 2018, more than 10% of hyperconverged integrated systems[1] deployments will suffer from avoidable network-induced performance problems, up from less than 1% of today's systems. It is also forecasted that 60% of providers will start offering integrated networking services, along with compute and storage services.

The costs of the networking infrastructure continue to raise at a time when the costs of the other components of the cloud infrastructure continue to decrease. Moreover, many cloud applications including analytics and applications in science and engineering are data-intensive and network-intensive and require more expensive networks with higher bandwidth and lower latency. The networking equipment represents about 8%, the servers account for 57%, and the power represents 31% [232] of the monthly costs of a CSP.

[1]Hyperconvergence is a software-centric architecture that tightly integrates compute, storage, networking, virtualization, and possibly other technologies in a commodity hardware box supported by a single vendor.

Cloud Computing. DOI: 10.1016/B978-0-12-812810-7.00007-8

This chapter is focused on communication and the discussion starts with an overview of the network used to access the cloud, the Internet, a packet-switched network of networks, and the World Wide Web in Sections 5.1, 5.2, and 5.3, followed by a discussion of Named Data Networks and Software Defined Networks in Sections 5.4 and 5.5, respectively. Then the focus is changed to the communication fabric used inside the cloud infrastructure and to the analysis of interconnection networks architecture and algorithms.

After an overview of the interconnection networks in Section 5.6 the chapter covers multistage networks, InfiniBand and Myrinet, and Storage Area Networks in Sections 5.7, 5.8, and 5.9, respectively. A scalable data center architecture and network resource management are the topics of Sections 5.10 and 5.11. Then the limelight changes again, this time to content delivery networks and vehicular networks in Sections 5.12 and 5.13. Further readings, historical notes, and a set of exercises and problems conclude the chapter.

5.1 PACKET-SWITCHED NETWORKS AND THE INTERNET

The Internet, a packet-switched network, provides access to computer clouds. A packet-switched network transports data units called *packets* through a maze of *switches* where packets are queued and routed towards their destination. Packets are subject to random delays, loss, and may arrive at their final destination out of order.

A few basic concepts are defined next. A *datagram* is a transfer unit in a packet-switched network. In addition to its *payload* a datagram has a header containing control information necessary for its transport through the network. A *network architecture* describes the protocol stack used for communication. A *protocol* is a set of rules on how to communicate, it specifies the actions taken by the sender and the receiver of a data unit. A network *host* identifies a system located at the network edge capable to initiate and to receive communication, be it a computer, a mobile device such as a phone, or a sensor.

Network architecture and protocols. A packet-switched network has a *network core* consisting of routers and control systems interconnected by very high bandwidth communication channels and a *network edge* where the end-user systems reside.

A packet-switched network is a complex system consisting of a very large number of autonomous components subject to complex and, sometimes, contradictory requirements. Basic strategies for implementing a complex system are *layering* and *modularization*. Layering means decomposing a complex function into elements interacting through well-defined channels, a layer can only communicate with its adjacent layers.

The protocol stack of the Internet, based on the TCP/IP network architecture is shown in Figure 5.1. At the sending host data flows down the protocol stack from the application layer to the transport layer, then to the network layer, and to the data link layer. The physical layer pushes the streams of bits through a physical communication link encoded either as electrical, optical, or electromagnetic signals. The corresponding data units for the five layer architecture are: messages, segments, packets, frames, and encoded bits, respectively.

The *transport layer* is responsible for end-to-end communication, from an application running on the sending host to its peer, running on the destination host using either TCP or UDP protocols. The network layer decides where the packet should be sent, either to another router, or to a destination

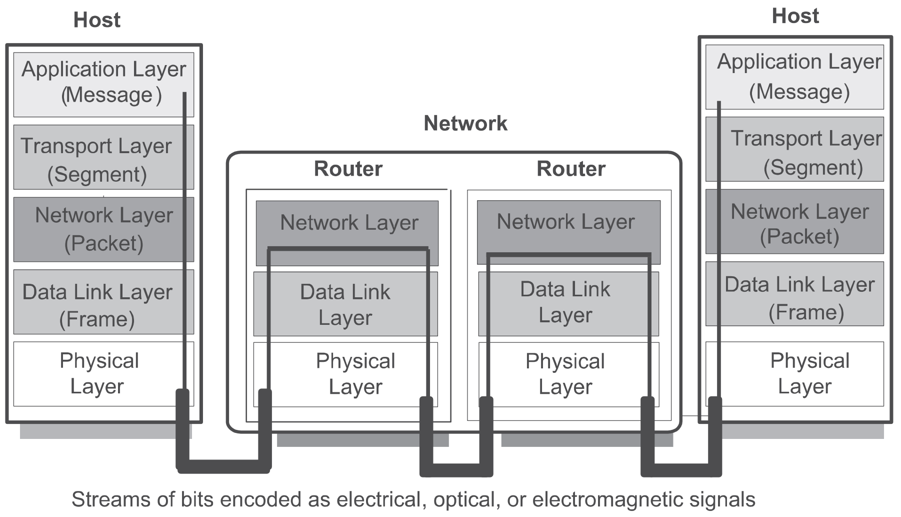


FIGURE 5.1

The Internet protocol stack. Applications running on hosts at the edge of the network communicate using application layer protocols. The transport layer deals with end-to-end delivery. The network layer is responsible for routing a packet through the network. The data link layer ensures reliable communication between adjacent nodes of the network, and the physical layer transports streams of bits encoded as electrical, optical, or electromagnetic signals (the thick lines represent such bit pipes).

host connected to a local area network connected to the router. IP, the *network layer* protocol, guides packets through the packet-switched network from the point of entry to the place where a packet exits the network. The *data link layer* encapsulates the packet for the communication link to the next hop. Once a packet reaches a router the bits are passed to the data link and then to the network layer.

A protocol on one system communicates with its *peer* on another system. For example, the transport protocol on the sender, host A, communicates with the transport protocol on the receiver, host B. On the sending side, A, the transport protocol encapsulates the data from the application layer and adds control information as headers that can only be understood by its peer, the transport layer on host B. When the peer receives the data unit, it carries out a decapsulation, retrieves the control information, removes the headers, then passes the payload to the next layer up, the application layer on host B.

The payload for the data link layer at the sending site includes the network header and the payload at the network layer. In turn, the network layer payload includes transport layer header and its payload consisting of the application layer header and application data.

The Internet. The Internet is a network of networks, a collection of separate, autonomous, and distinct networks. All networks adhere to a common framework and use: (i) globally unique IP addresses; (ii) the IP (Internet Protocol) routing protocol; and (iii) the Border Gateway Routing (BGP) protocol. BGP is a path vector reachability protocol making core routing decisions. BGP maintains a table of IP networks designating network reachability among autonomous systems. BGP makes routing decisions based on path, network policies, and/or rule sets.

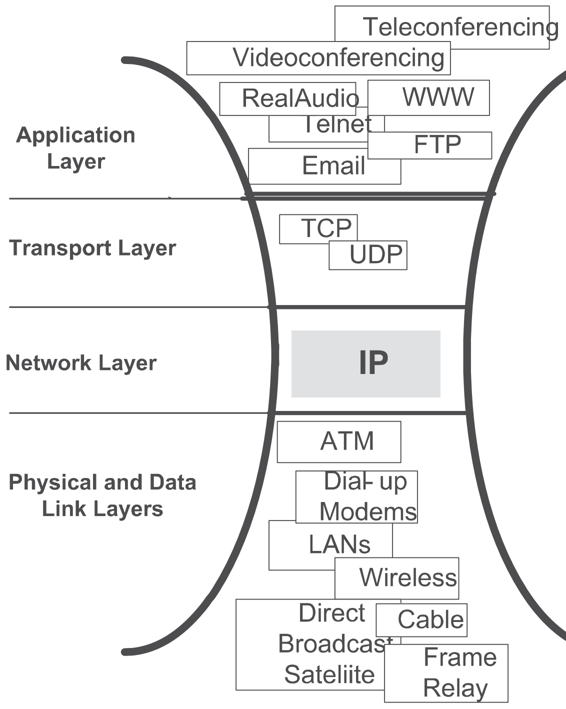


FIGURE 5.2

The hourglass network architecture of the Internet. Regardless of the application, the transport protocol, and the physical network, all packets are routed from the source to the destination using the IP protocol and the IP address of the destination.

An *IP address* is a string of integers uniquely identifying every host connected to the Internet. An IP address allows the network to identify first the destination network and then the host in that network where a datagram should be delivered. A host may have multiple IP addresses and it may be connected to more than one network. A host could be a supercomputer, a workstation, a laptop, a mobile phone, a network printer, or any other physical device with a network interface.

The Internet is based on a hourglass network architecture, see Figure 5.2. *The hourglass architecture is partially responsible for the explosive growth of the Internet*, it allowed the lower layers of the architecture to evolve independently from the upper layers. The communication technology drives the dramatic change of the lower layers of the Internet architecture including the increase of the communication channels bandwidth, the widespread use of wireless networks, and of satellite communication. The software and the applications are the engines of progress for the upper layers of the architecture.

The hourglass model reflects the *end-to-end* architectural design principle. The model captures the fact that all packets transported through the Internet use IP to reach their destination. IP only provides *best effort delivery*. Best effort delivery means that any router along the path from the source to the destination may drop a packet when it is overloaded.

Another important architectural design principle of the Internet is the *separation between the forwarding and the routing planes.* The *forwarding plane* decides what to do with packets arriving on an inbound interface of a router. The plane uses a table to lookup the destination address of an incoming packet, then it retrieves the information to determine the path from the receiving element to the proper outgoing interface(s) through the internal forwarding fabric of the router. The *routing plane* is responsible for building the routing table in each router. The separation of the two planes allowed the forwarding plane to function, while the routing evolved.

In addition to the IP or logical address, each network interface, the hardware connecting a host with a network, has a unique *physical* or *MAC address*. While the MAC address is permanently assigned to a network interface of the device, the IP address may be dynamically assigned. The IP address of a mobile device changes depending on the device location and the network it is connected to.

The Dynamic Host Configuration Protocol (DHCP) is an automatic configuration protocol. DHCP assigns an IP address to a client system. A DHCP server has three methods of allocating IP addresses:

1. Dynamic allocation – a network administrator assigns a range of IP addresses to DHCP. During network initialization each client computer on the LAN is configured to request an IP address from the DHCP server. The request-and-grant process uses a lease concept with a controllable time period, allowing the DHCP server to reclaim (and then reallocate) IP addresses that are not renewed.
2. Automatic allocation – the DHCP server permanently assigns a free IP address to a client, from the range defined by the administrator.
3. Static allocation – the DHCP server allocates an IP address based on a manually filled in table with (MAC address – IP address) pairs. Only a client with a MAC address listed in this table is allocated an IP address.

Once a packet reaches the destination host it is delivered to the proper transport protocol daemon which, in turn, delivers it to the application which listens to an abstraction of the end-point of a logical communication channel called a *port*, Figure 5.3. The processes or threads running an application use an abstraction called *socket* to send and receive data through the network. A socket manages one queue of incoming messages and another one for outgoing messages.

Internet transport protocols. The Internet uses two transport protocols, a connectionless datagram protocol, UDP (User Datagram Protocol) and a connection-oriented protocol, TCP (Transport Control Protocol). The header of a datagram contains information sufficient for routing through the network from the source to the destination. The arrival time and order of delivery of datagrams are not guaranteed.

To ensure efficient communication, the UDP transport protocol assumes that error checking and error correction are either not necessary or performed by the application. Datagrams may arrive out of order, duplicated, or may not arrive at all. Applications using UDP include: the DNS (Domain Name System), VoIP, TFTP (Trivial File Transfer Protocol), streaming media applications such as IPTV, and online games.

TCP provides reliable, ordered delivery of a stream of bytes from an application on one system to its peer on the destination system. An application sends/receives data units called *segments* to/from a specific port, an abstraction of and end-point of a logical communication link. TCP is the transport protocol used by the World Wide Web, email, file transfer, remote administration, and many other important applications.

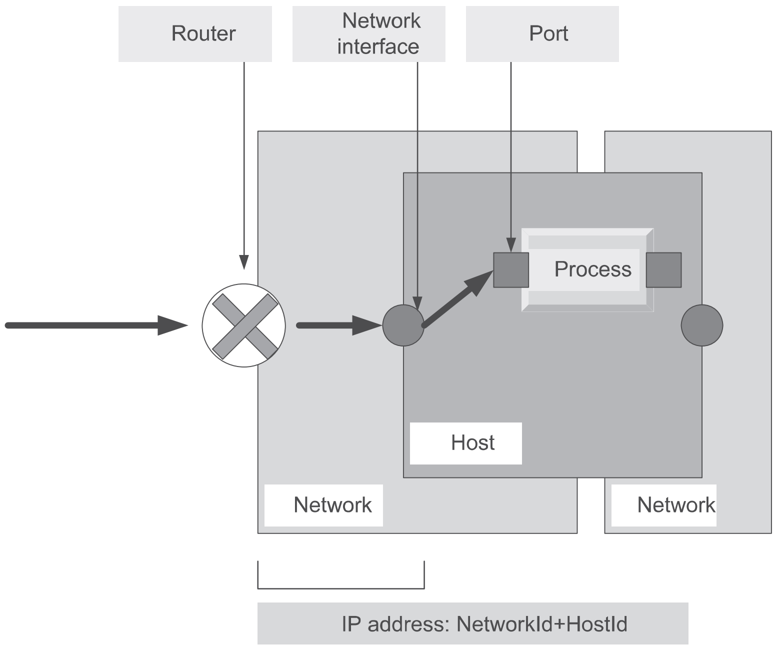


FIGURE 5.3

Packet delivery to processes and threads; a packet is first routed by the IP protocol to the destination network and then to the host specified by the IP address. Applications listen to *ports*, abstractions of the end point of a communication channel.

TCP uses an end-to-end *flow control mechanism* based on a sliding-window, a range of packets the sender can send before receiving an acknowledgment from the receiver. This mechanisms allows the receiver to control the rate of segments sent and process them reliably.

A network has a finite capacity to transport data and when its load is approaching this capacity, we witness undesirable effects, the routers start dropping packets, the delays and the jitter increase. An obvious analogy is a highway where the time to travel from point A to point B increases dramatically in case of congestion. A solution for traffic management is to introduce traffic lights limiting the rate at which new traffic is allowed to enter the highway and this is precisely what the TCP emulates.

TCP uses several mechanisms for *congestion control*, see Section 5.3. These mechanisms control the rate of the data entering the network, keeping the data flow below a rate that would lead to a network collapse and enforcing a fair allocation among flows. Acknowledgments coupled with timers are used to infer network conditions between the sender and receiver.

TCP congestion control policies are based on four algorithms, *slow-start, congestion avoidance, fast retransmit*, and *fast recovery*. These algorithms use local information, such as the RTO (retransmission timeout) based on the estimated RTT (round-trip time) between the sender and receiver, as well as the variance in this round trip time to implement the congestion control policies. UDP is a connectionless protocol thus, there are no means to control the UDP traffic.

The review of basic networking concepts in this section shows why process-to-process communication incurs a significant overhead. While raw speed of fiber optic channels can reach Tbps,[2] the actual transmission rate for end-to-end communication over a wide area network can only be of the order of tens of Mbps and the latency is of the order of milliseconds. This has important consequences for the development of computer clouds. The term "speed" is used informally to describe the maximum data transmission rate, or the capacity of a communication channel; this capacity is determined by the physical bandwidth of the channel and this explains why the term channel "bandwidth" is also used to measure the channel capacity, or the maximum data rate.

5.2 THE TRANSFORMATION OF THE INTERNET

The Internet is continually evolving under the pressure of its own success and the need to accommodate new applications and a larger number of users. Initially conceived as a data network, a network designed to transport data files, the Internet has morphed into today's network supporting data-streaming and applications with real-time constraints such as the Lambda service offered by the AWS. The discussion in this section is restricted to the aspects of the Internet evolution relevant to cloud computing.

Tier 1, 2, and 3 networks. To understand the architectural consequences of Internet evolution we discuss first the relation between two networks. *Peering* means that two networks exchange traffic between each other's customers freely. *Transit* requires a network to pay another one for accessing the Internet. The term *customer* means that a network is receiving money to allow Internet access.

Based on these relations the networks are commonly classified as Tier 1, 2, and 3. A *Tier 1 network* can reach every other network on the Internet without purchasing IP transit or paying settlements; examples of Tire 1 networks are Verizon, ATT, NTT, Deutsche Telecom, see Figure 5.4.

A *Tier 2 network* is an Internet service provider who engages in the practice of peering with other networks, but who still purchases IP transit to reach some portion of the Internet; Tier 2 providers are the most common providers on the Internet. A *Tier 3 network* purchases transit rights from other networks (typically Tier 2 networks) to reach the Internet. A *point-of-presence (POP)* is an access point from one place to the rest of the Internet.

An *Internet exchange point* (IXP) is a physical infrastructure allowing *Internet Service Providers* (ISPs) to exchange Internet traffic. IXPs interconnect networks directly, via the exchange, rather than through one or more third party networks. The advantages of the direct interconnection are numerous, but the primary reasons to implement an IXP are cost, latency, and bandwidth. Traffic passing through an exchange is typically not billed by any party, whereas traffic to an ISP's upstream provider is.

IXPs reduce the portion of an ISP's traffic which must be delivered via their upstream transit providers, thereby reducing the average per-bit delivery cost of their service. Furthermore, the increased number of paths found through the IXP improves routing efficiency and fault-tolerance. A typical IXP consists of one or more network switches, to which each of the participating ISPs connects.

New technologies such as web applications, cloud computing, and content-delivery networks are reshaping the definition of a network as we can see in Figure 5.5 [287]. The World Wide Web, gam-

[2]NTT (Nippon Telegraph and Telephone) achieved a speed of 69.1 Tbps in 2010 using wavelength division multiplexing of 432 wavelengths with a capacity of 171 Gbps over a 240 km-long optical fiber.

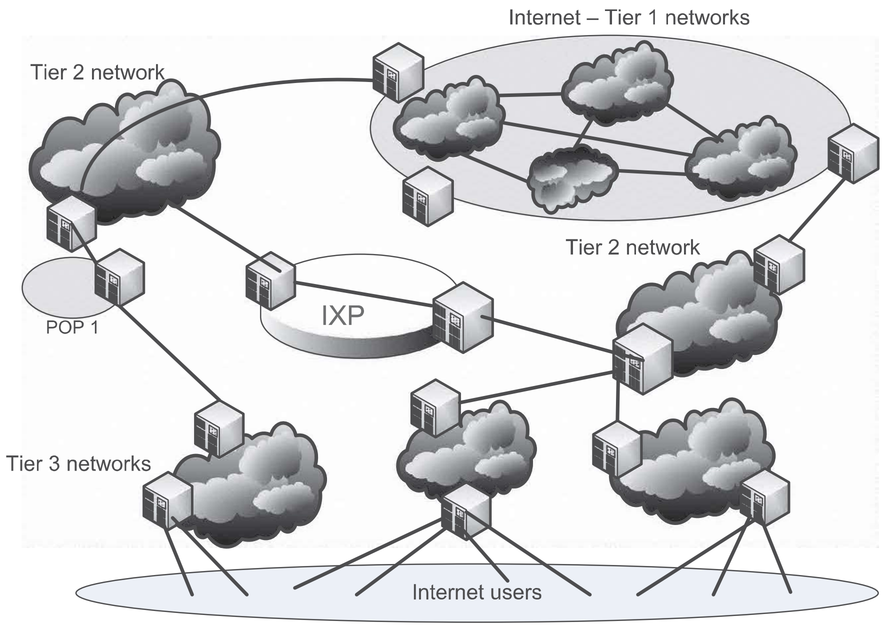


FIGURE 5.4

The relation of Internet networks based on the transit and paying settlements. There are three classes of networks, Tier 1, 2, and 3; an IXP is a physical infrastructure allowing ISPs to exchange Internet traffic.

ing, and entertainment are merging and more computer applications are moving to the cloud. Data streaming consumes an increasingly larger fraction of the available bandwidth as high definition TV sets become less expensive and content providers such as Netflix and Hulu offer customers services that require a significant increase of the network bandwidth.

Does the network infrastructure adequately respond to the current demand for bandwidth? The Internet infrastructure in the US is falling behind in terms of network bandwidth, see Figure 5.6. A natural question to ask is: Where is the actual bottleneck limiting the bandwidth available to a typical Internet broadband user? The answer is: the "last mile," the link connecting the home to the ISP network. Recognizing that the broadband access infrastructure ensures continual growth of the economy and allows people to work from any site, Google has initiated the Google Fiber Project which aims to provide a one Gbps access speed to individual households through FTTH.[3]

Migration to IPv6. The Internet Protocol, Version 4 (IPv4), provides an addressing capability of 2^{32}, or approximately 4.3 billion addresses, a number that proved to be insufficient. Indeed, the Internet Assigned Numbers Authority (IANA) assigned the last batch of 5 address blocks to the Regional In-

[3]The fiber-to-the-home (FTTH) is a broadband network architecture that uses optical fiber to replace the copper-based local loop used for the last mile network access to home.

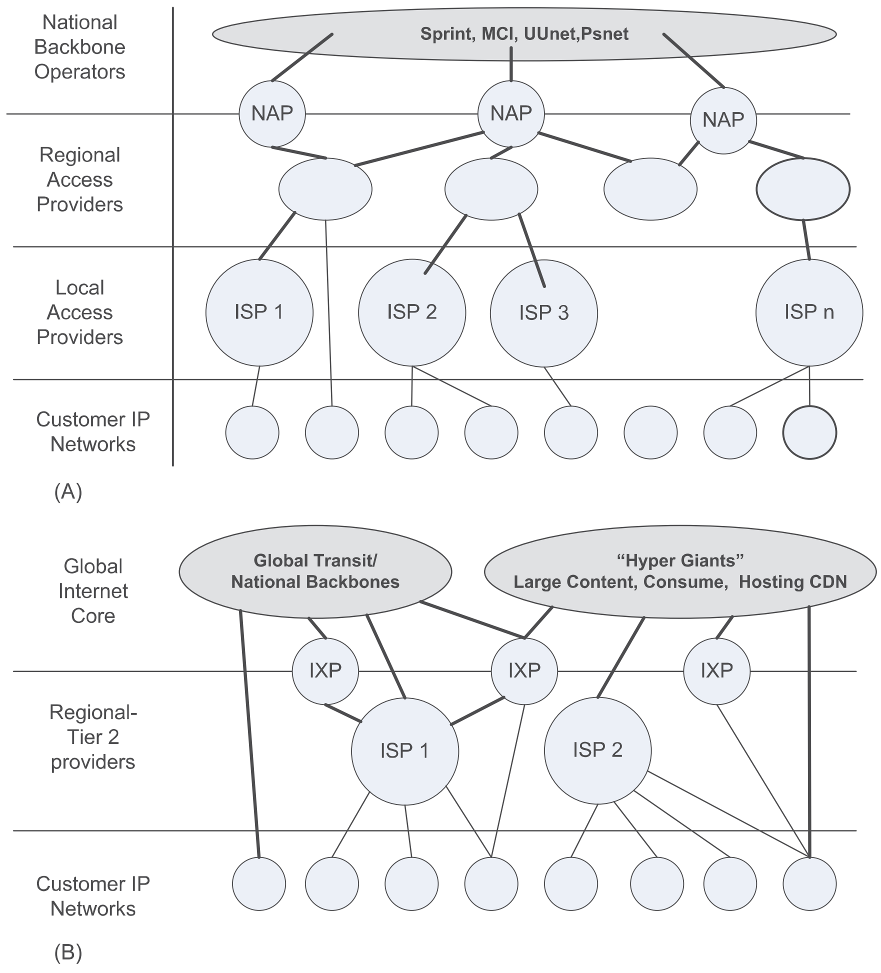

National Backbone Operators
Sprint, MCI, UUnet,Psnet
NAP
NAP
NAP
Regional Access Providers
Local Access Providers
ISP 1
ISP 2
ISP 3
ISP n
Customer IP Networks
(A)
Global Internet Core
Global Transit/ National Backbones
"Hyper Giants" Large Content, Consume, Hosting CDN
IXP
IXP
IXP
Regional- Tier 2 providers
ISP 1
ISP 2
Customer IP Networks
(B)

FIGURE 5.5

The transformation of the Internet; the traffic carried by Tier 3 networks increased from 5.8% in 2007 to 9.4% in 2009; Goggle applications accounted for 5.2% of the traffic in 2009 [287].

ternet Registries in February 2011, officially depleting the global pool of completely fresh blocks of addresses; each of the address blocks represents approximately 16.7 million possible addresses.

The Internet Protocol, Version 6 (IPv6), provides an addressing capability of 2^{128}, or 3.4×10^{38} addresses. There are other major differences between IPv4 and IPv6:

- *Multicasting.* IPv6 does not implement traditional IP broadcast, i.e. the transmission of a packet to all hosts on the attached link using a special broadcast address and, therefore, does not define broadcast addresses. IPv6 supports new multicast solutions, including embedding rendezvous point

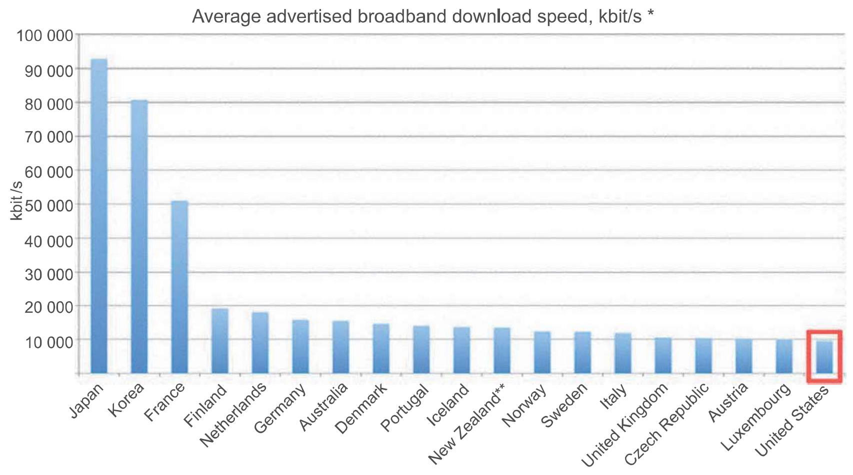


FIGURE 5.6

The broadband access, the average download speed advertised by several countries.

addresses in an IPv6 multicast group address. This solution simplifies the deployment of inter-domain solutions.

- *Stateless address autoconfiguration (SLAAC).* IPv6 hosts can configure themselves automatically when connected to a routed IPv6 network using the Internet Control Message Protocol version 6 (ICMPv6) router discovery messages. When first connected to a network, a host sends a link-local router solicitation multicast request for its configuration parameters. If suitably configured, routers respond to such a request with a router advertisement packet that contains network-layer configuration parameters.
- *Mandatory support for network security.* Internet Network Security (IPsec) is an integral part of the base protocol suite in IPv6 while it is optional for IPv4. IPsec is a protocol suite operating at the IP layer. Each IP packet is authenticated and encrypted. Other security protocols, e.g., the Secure Sockets Layer (SSL), the Transport Layer Security (TLS) and the Secure Shell (SSH) operate at the upper layers of the TCP/IP suite. IPsec uses several protocols: (1) Authentication Header (AH) supports connectionless integrity, data origin authentication for IP datagrams, and protection against replay attacks; (2) Encapsulating Security Payload (ESP) supports confidentiality, data-origin authentication, connectionless integrity, an anti-replay service, and limited traffic-flow confidentiality; (3) Security Association (SA) provides the parameters necessary to operate the AH and/or ESP operations.

Unfortunately, migration to IPv6 is a very challenging and costly proposition [115]. A simple analogy allows us to explain the difficulties related to migration to IPv6. The telephone numbers in North America consist of 10 decimal digits. This scheme supports up to 10 billion phones but, in practice, we have fewer available numbers. Indeed, some phone numbers are wasted because we use area codes based on geographic proximity and, on the other hand not all available numbers in a given area are allocated.

Table 5.1 Web statistics collected from a sample of several billion pages detected during Google's crawl and indexing pipeline.

Metric	Value
Number of sample pages analyzed	4.2×10^9
Average number of resources per page	44
Average number of GETs per page	44.5
Average number of unique host names encountered per page	7
Average size transferred over the network per page, including HTTP headers	320 KB
Average number of unique images per page.	29
Average size of the images per page	206 KB
Average number of external scripts per page	7
Number of sample SSL (HTTPS) pages analyzed	17×10^6

To overcome the limited number of phone numbers in this scheme, large organizations use private phone extensions that are typically 3 to 5 digits long; thus, a single public phone number can translate to 1000 phones for an organization using a 3 digit extension. Analogously, Network Address Translation (NAT) allow a single public IP address to support hundreds or even thousands of private IP address. In the past NAT did not work well with applications such as VoIP (Voice over IP) and VPN (Virtual Private Network). Nowadays Skype and STUN VoIP applications work well with NAT. Now NAT-T and SSLVPN support VPN NAT.

If the telephone companies decide to promote a new system based on 40 decimal digit phone numbers we will need new telephones. At the same time we will need new phone books, much thicker as each phone number is 40 characters instead of 10, each individual needs a new personal address book, and virtually all the communication and switching equipment and software need to be updated. Similarly, the IPv6 migration involves upgrading all applications, hosts, routers, and DNS infrastructure; also, moving to IPv6 requires backward compatibility, any organization migrating to IPv6 should maintain a complete IPv4 infrastructure.

5.3 WEB ACCESS AND THE TCP CONGESTION CONTROL WINDOW

The web supports access to content stored on a cloud. Virtually all cloud computing infrastructures allow users to interact with their computations on the cloud using web-based systems. It should be clear that the metrics related to web access are important for designing and tuning the networks. The site http://code.google.com/speed/articles/web-metrics.html provides statistics about metrics such as the size and the number of resources and Table 5.1 summarizes these metrics. The statistics are collected from a sample of several billion pages detected during Google's crawl and indexing pipeline.

Such statistics are useful to tuning the transport protocols to deliver optimal performance in terms of latency and throughput. Metrics, such as the average size of a page, the number of GET operations, are useful to explain the results of performance measurements carried out on existing systems and to propose changes to optimize the performance as discussed next. HTTP, the application protocol for web browsers uses TCP and takes advantage of its congestion control mechanisms.

TCP flow control and congestion control. To achieve high channel utilization, avoid congestion, and, at the same time, ensure a fair sharing of the network bandwidth TCP uses two mechanisms, flow control and congestion control. A *flow control* mechanism throttles the sender, feedback from the receiver forces the sender to transmit only the amount of data the receiver is able to buffer and then process. TCP uses a sliding window flow control protocol. If W is the window size, then the data rate S of the sender is:

$$S = \frac{W \times MSS}{RTT} \text{ bps} = \frac{W}{RTT} \text{ packets/second} \tag{5.1}$$

where MSS and RTT denote the Maximum Segment Size and the Round Trip Time, respectively, assuming that MMS is 1 packet. If S is too small the transmission rate is smaller than the channel capacity, while a large S leads to congestion. The channel capacity depends on the network load as the physical channels along the path of a flow are shared with many other Internet flows.

The actual window size W is affected by two factors: (a) the ability of the receiver to accept new data and (b) the sender's estimation of the available network capacity. The receiver specifies the amount of additional data it is willing to accept in the *receive window* field of every frame. The receiver's window shifts when the receiver receives and acknowledges a new segment of data. When a receiver advertises a window size of zero, the sender stops sending data and starts the persist timer. This timer is used to avoid the deadlock when a subsequent window size update from the receiver is lost.

When the persist timer expires, the sender sends a small packet and the receiver responds by sending another acknowledgment containing the new window size. In addition to the flow control provided by the receiver, the sender attempts to infer the available network capacity and to avoid overloading the network. The source uses the losses and the delay to determine the level of congestion. If $awnd$ denotes the receiver window and $cwnd$ the congestion window set by the sender, the actual window should be:

$$W = \min(cwnd, awnd). \tag{5.2}$$

Several algorithms are used to calculate $cwnd$ including Tahoe and Reno, developed by Van Jacobson in 1988 and 1990. Tahoe was based on slow start (SS), congestion avoidance (CA), and fast retransmit (FR). The sender probes the network for spare capacity and detects congestion based on loss. The slow start means that the sender starts with a window of two times MSS, $init\_cwnd = 1$. For every packet acknowledged, the congestion window increases by 1 MSS so that the congestion window effectively doubles for every RTT.

When the congestion window exceeds the threshold, $cwnd \geq ssthresh$, the algorithm enters the CA state. In CA state, on each successful acknowledgment $cwnd \leftarrow cwnd + 1/cwnd$ and on each RTT $cwnd \leftarrow cwnd + 1$. The fast retransmit is motivated by the fact that the timeout is too long thus, a sender retransmits immediately after 3 duplicate acknowledgments without waiting for a timeout. Two adjustments are made in this case:

$$flightsize = \min(awnd, cwnd) \quad \text{and} \quad ssthresh \leftarrow \max(flightsize/2, 2) \tag{5.3}$$

and the system enters in the slow start state, $cwnd = 1$. The pseudocode describing the Tahoe algorithm is:

```
for every ACK {
        if (W < ssthresh) then W++      (SS)
        else    W += 1/W                (CA)
    }
    for every loss {
        ssthresh = W/2
            W  = 1
    }
```

The pattern of usage of the Internet has changed. Measurements reported by different sources [156] show that in 2009 the average bandwidth of an Internet connection was 1.7 Mbps. More than 50% of the traffic required more than 2 Mbps and could be considered broadband, while only about 5% of the flows required less that 256 Kbps and could be considered narrowband. Recall that the average web page size is in the range of 384 KB.

While the majority of the Internet traffic is due to long-lived, bulk data transfer, e.g., video and audio streaming, the majority of transactions are short-lived, e.g., web requests. So a major challenge is to ensure some fairness for short-lived transactions.

To overcome the limitations of the slow start, application strategies have been developed to reduce the time to download data over the Internet. For example, two browsers, Firefox 3 and Google Chrome open up to six TCP connections per domain to increase the parallelism and to boost start-up performance when downloading a web page. The Internet Explorer 8 opens 180 connections. Clearly, these strategies circumvent the mechanisms for congestion control and incur a considerable overhead. It is argued that a better solution is to increase the initial congestion window of TCP and the arguments presented in [156] are:

- The TCP latency is dominated by the number of RTT's during the slow start phase. Increasing the $init\_cwnd$ parameter allows the data transfer to be completed with fewer RTT's.
- Given that the average page size is 384 KB, a single TCP connection requires multiple RTT's to download a single page.
- It ensures fairness between short-lived transactions which are a majority of Internet transfers and the long-lived transactions which transfer very large amounts of data.
- It allows faster recovery after losses through Fast Retransmission.

It can be shown that the latency of a transfer completing during the slow start without losses is given by the expression

$$\left\lceil \log_\gamma \left(\frac{L(\gamma - 1)}{init\_cwnd} + 1 \right) \right\rceil \times RTT + \frac{L}{C} \tag{5.4}$$

with L the transfer size, C the bottleneck-link rate, and γ a constant equal to 1.5 or 2 depending if the acknowledgments are delayed or not; $L/init\_cwnd \geq 1$. In the experiments reported in [156] the TCP latency was reduced from about 490 msec when $init\_cwnd = 3$ to about 466 msec for $init\_cwnd = 16$.

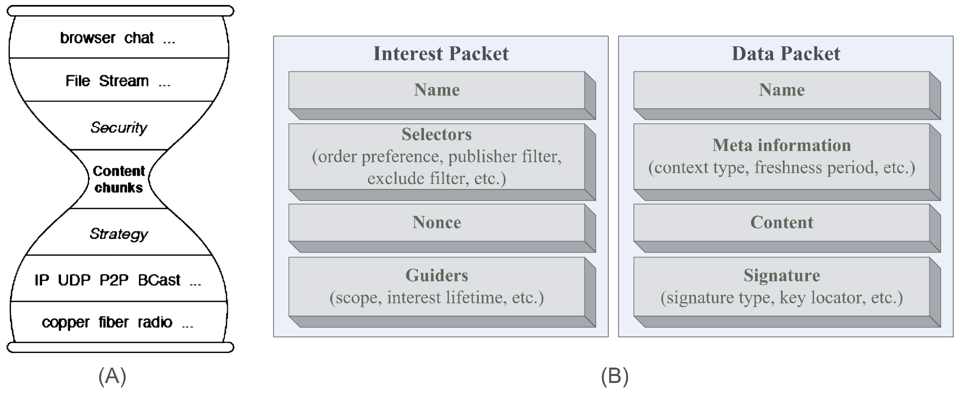


FIGURE 5.7

(A) The NDN hourglass architecture parallels the one of the Internet, it separates the lower layers of the protocol stack from the upper ones thus, naming of the data can evolve independently from networking. (B) The semantics of the NDN networking service is to fetch a data chunk identified by name, while the Internet semantics is to deliver a packet to a given network address through an end-to-end channel identified by the source and the destination IP addresses. Two packet types, Interest and Data, see http://named-data.net/doc/ndn-tlv/ are used for NDN routing and forwarding.

5.4 NAMED DATA NETWORKS

The Internet is a network of *communication networks* where the communication units, the packets, only name the communication end points. Today's Internet is mostly used as a *distribution network* by applications in areas such as digital media, electronic commerce, Big Data analytics, and so on.

In a distribution network communication is *content-centric*, named objects are requested by an agent and, once a site holding the object is located, the network transports it to the site that requested the object. The end user in a distribution network is oblivious to the location of the data and it is only interested in the content. The data in today's communication networks is tied to a particular host and this makes data replication and migration difficult.

The idea of a content-centric network has been around for some time. In 1999 the TRIAD project at Stanford[4] [205,487] proposed to use the name of an object to route towards a replica of it. In this proposal the Internet Relay Protocol performs name-to-address conversion using the routing information maintained by relay nodes. The Name-Based Routing Protocol performs a function similar to the BGP protocol, it supports a mechanism for updating the routing information in relay nodes.

In 2006 the Data Oriented Network Architecture (DONA) project at U.C. Berkeley [150] extended TRIAD incorporating security and persistence as primitives of the new network architecture. DONA architecture exposes two primitives: FIND – allows a client to request a particular piece of data by its name and REGISTER – enables content providers to indicate their intent to serve a particular data

[4]TRIAD stands for Translating Relaying Internet Architecture Integrating Active Directories.

object. In 2006 Van Jacobson from UCLA argued that Named Data Networks (NDNs) should be the architecture of the future Internet.

In 2012 the Internet Research Task Force (IRTF) established an information-centric networking (ICN) research working group for investigating architecture designs for NDN. A 2014 survey of ICN research and a succinct presentation of NDNs can be found in [532] and [548], respectively. The important features of the NDN architecture addressed by the current research efforts include namespaces, trust models, in-network storage, data synchronization, and last but not least, rendezvous, discovery, and bootstrapping.

The hourglass model can be extended, the packets can name objects rather than communication endpoints. The NDNs hourglass model separates the lower layers of the protocol stack from the upper ones thus, data naming can evolve independently from networking, see Figure 5.7A.

The NDN packet delivery is driven by the data consumers, the communication is initiated by an agent generating an *Interest* packet containing the name of the data, see Figure 5.7B. Once the Interest packet reaches a network host which has a copy of the data item, a *Data* packet containing the name, the data contents, and the signature is generated. The signature consists of the producer's key. The *Data* packet follows the route traced by the *Interest* packet and it is delivered to the data consumer agent. An NDN router maintains the information necessary to forward the packet:

1. *Content Store* – local cache for *Data* packets previously crossing the router. When an Interest packet arrives, a search of the content store determines if a matching data exists and if so the data is forwarded on the same router interface the Interest packet was received. If not, the router uses a data structure, the Forwarding Information Base, to forward the packet. More recently, NDN supports more persistent and larger-volume in-network storage, called Repositories providing services similar to that of the Content Delivery Networks.
2. *Forwarding Information Base* – the entries of this data structure are populated by a name-prefix based procedure. The Forwarding Strategy retrieves the longest prefix matched entry from forwarding information base for an Interest packet.
3. *Pending Interest Table* (PIT) – stores all the Interest packets the router has forwarded but have not been satisfied yet. A PIT entry records the data name carried in the Internet, together with its incoming and outgoing router interface(s). When a Data packet arrives, the router finds the matching PIT entry and forwards the data to all downstream interfaces listed in that PIT entry, then removes the PIT entry, and caches the Data packet in the Content Store.

Names are essential in NDN, though namespace management is not part of the NDN architecture. The scope and contexts of NDN names are different. Globally accessible data must have globally unique names, while local names require only local routing and must only be locally unique. There is no namespace exhaustion in NDN, while migration to IPv6 has become a necessity due to the IP address limitations of IPv4.

There are other important differences between NDN and TCP/IP. Each *Data* packet is cryptographically signed thus the system supports data-centric security, while TCP/IP security is left to the communication endpoints. An NDN router announces name prefixes for the data it is willing to serve, while an IP router announces IP prefixes.

NDN is a universal overlay,[5] it can run over any datagram network and, conversely, any datagram network, including IP, can run over NDN. Classical algorithms such as Open Shortest Path First (OSPF) route data chunks using component-wise longest prefix match of the name in an Interest packet with the FIB entries of a router.

Wide area applications can operate over IP tunnels and islands of NDN nodes could be interconnected by tunneling over non-NDN clouds. Scalable forwarding along with robust and effective trust management are critical challenges for the future of NDN networks. Namespace design is at the heart of NDNs as it involves application data, communication, storage models, routing, and security.

NDN could be useful for cloud data centers and in particular for those supporting the IaaS cloud delivery model. Data is replicated and multiple instances could access data concurrently from their nearest storage servers. This would be useful for some MapReduce applications but it would require major changes in the frameworks supporting this paradigm. On the other hand, many applications cache data in the server memory and they would only marginally benefit from the NDN support.

5.5 SOFTWARE DEFINED NETWORKS

Software-defined Networks (SDN) extend basic principles of resource virtualization to networking to allow programmatic control of communication networks. SDNs introduce an abstraction layer separating network configuration from the physical communication resources. More precisely, a network operating system running inside a control layer, sandwiched between the application layer and the infrastructure layer, allows applications to re-configure dynamically the communication substrate to adapt to their security, scalability, and manageability needs.

Though enthusiastically embraced by networking vendors, the SDN technology is largely conceptual and there is little agreement either on the architecture, the APIs, or the overlay networks among vendors. A 2012 white paper of the Open Network Foundation (ONF) (https://www.opennetworking.org/sdn-resources) promoted OpenFlow-based SDNs. According to ONF a new network architecture is necessary for several reasons including changing traffic patterns due to several factors such as the rise of cloud services and the need for more bandwidth demanded by Big Data applications. The ONF architecture includes several components:

- Applications – generate network requirements to controllers.
- Controllers – translate application requirements to SDN datapaths and provide the applications with an abstract view of the network.
- Datapaths – logical network devices which expose control to the physical substrate; consist of agents and forwarding engines.
- Control-to-Data-Plane Interfaces – the interfaces between SDN controllers and SDN datapaths.
- Northbound Interfaces – interfaces between applications and controllers.
- Interface Drivers and Agents – driver-agent pairs.
- Management and Administration.

[5]An overlay network is a network built on top of another physical network, its nodes are connected by virtual links, each of which corresponds to a path, possibly through many physical links, in the underlying network.

OpenFlow is an API for programming data plane switches. The data path and the control paths of an OpenFlow switch consist of a flow table including an action associated with each flow entry and a controller which programs the flow entry. The controller configures and manages the switch and receives events from the switch.

5.6 INTERCONNECTION NETWORKS FOR COMPUTER CLOUDS

Interconnection networks for multiprocessor systems, supercomputers, and cloud computing are discussed in the next sections. Computing and communication are deeply intertwined as we have seen in Chapters 3 and 4 and interconnection networks are critical for the performance of computer clouds and supercomputers.

Several concepts important for understanding interconnection networks are introduced next. A network consists of nodes and links or communication channels. The *degree* of a node is the number of links the node is connected to. The *nodes* of a interconnection network could be processors, memory units, or servers. The *network interface* of a node is the hardware connecting it to the network.

Switches and *communication channels* are the elements of the interconnection fabric. Switches receive data packets, look inside each packet to identify the destination IP addresses, then use the routing tables to forward the packet to the next hop towards its final destination. An *n-way switch* has *n* ports that can be connected to *n* communication links. An interconnection network can be *non-blocking* if it is possible to connect any permutation of sources and destinations at any time. An interconnection network is *blocking* if this requirement is not satisfied.

While processor and memory technology have followed Moore's Law, interconnection networks have evolved at a slower pace and have become a major factor in determining the overall performance and cost of the system. For example, from 1997 to 2010 the speed of the ubiquitous Ethernet network has increased from 1 to 100 Gbps. This increase is slightly slower than the Moore's Law for traffic [354] which predicted, 1 Tbps Ethernet by 2013.

Interconnection networks are distinguished by their topology, routing, and flow control. *Network topology* is determined by the way nodes are interconnected, routing decides how a message gets from its source to destination, and the flow control negotiates how the buffer space is allocated. There are two basic types of network topologies:

- Static networks where there are direct connections between servers;
- Switched networks where switches are used to interconnect the servers.

The topology of an interconnection network determines the *network diameter*, the average distance between all pairs of nodes, the *bisection width*, the minimum number of links cut to partition the network into two halves, the bisection bandwidth, as well as the cost and the power consumption [271]. When a network is partitioned into two networks of the same size the bisection bandwidth measures the communication bandwidth between the two.

The *full bisection bandwidth* allows one half of the network nodes to communicate simultaneously with the other half of the nodes. Assume that half of the nodes inject data into the network at a rate *B* Mbps. When the bisection bandwidth is *B* then the network has full bisection bandwidth. The switching fabric must have sufficient bi-directional bandwidth for cloud computing.

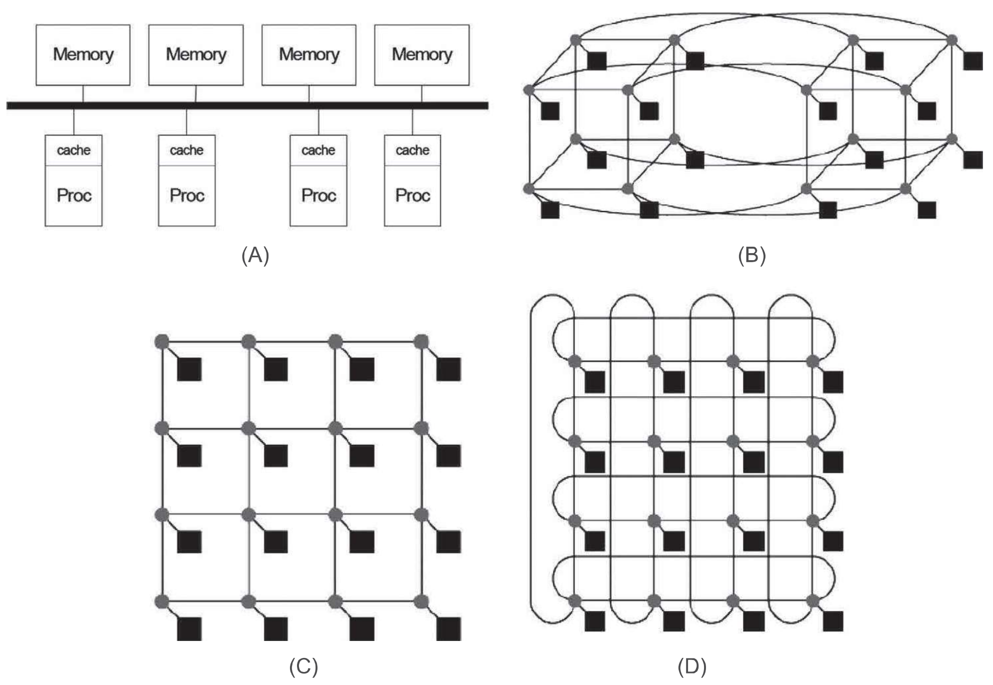


FIGURE 5.8

Static interconnection networks. (A) Bus; (B) Hypercube; (C) 2D-mesh: (D) 2D-torus.

Some of the most popular topologies with a static interconnect are:

- Bus, a simple and cost-effective network, see Figure 5.8A. It does not scale, but it is easy to implement cache coherence through snooping for distributed memory systems. A bus is often used in shared memory multiprocessor systems.
- Hypercube of order n, see Figure 5.8B. A hypercube has a good bisection bandwidth, the number of nodes is $N = 2^n$, the degree is $n = \log N$, and the average distance between nodes is $\mathcal{O}(N)$ hops. Example of use: SGI Origin 2000.
- 2-D mesh, see Figure 5.8C. An $n \times n$ 2D-mesh has many paths to connect nodes, has a cost of $\mathcal{O}(n)$, and the average latency is $\mathcal{O}(\sqrt{n})$. A mesh is not symmetric on edges thus, its performance is sensitive to the placement of communicating nodes on edges versus the middle. Example of use: Intel Paragon supercomputer of the 1990s.
- Torus, avoids the asymmetry of the mesh, but has a higher cost in terms of the number of components. A torus is good for applications using nearest-neighbor communication, see Figure 5.8D. It is prevalent for proprietary interconnects. Example of use: 6-D Mesh/torus of Fujitsu K supercomputer.

Switched networks have multiple layers of switches connecting the nodes as shown in Figure 5.9:

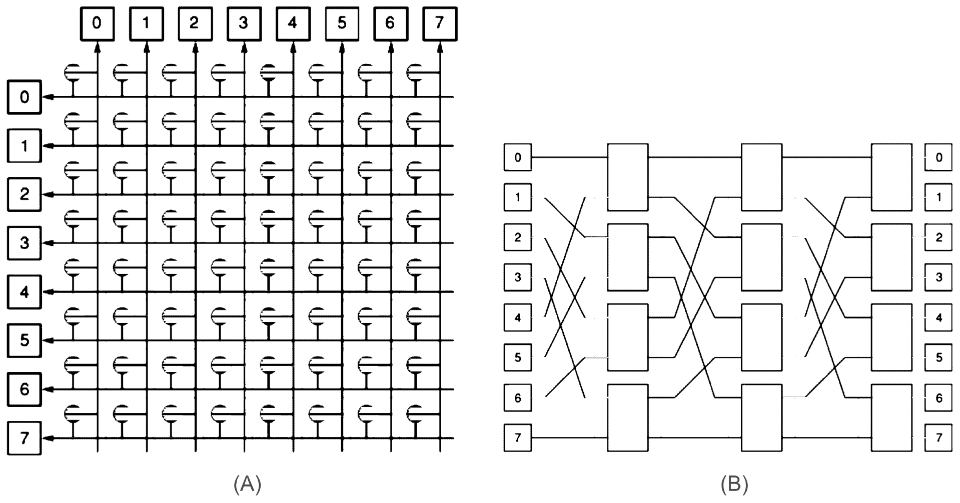


FIGURE 5.9

Switched networks. (A) An 8 × 8 crossbar switch. 16 nodes are interconnected by 49 switches represented by the dark circles; (B) An 8 × 8 Omega switch. 16 nodes are interconnected by 12 switches represented by white rectangles.

- Crossbar switch – has N^2 crosspoint switches, see Figure 5.9A
- Omega (Butterfly, Benes, Banyan, etc.) have $(N \log N)/2$ switches, see Figure 5.9B. The cost is $\mathcal{O}(N \log N)$ and the latency is $\mathcal{O}(\log N)$. Omega networks are low diameter networks.

Cloud interconnection networks. The cloud infrastructure consists of one or more Warehouse Scale Computers (WSCs) discussed in Section 8.2. A WSC has a hierarchical organization with a large number of servers interconnected by high-speed networks.

The networking infrastructure of a cloud must satisfy several requirements including scalability, cost, and performance. The network should allow low-latency, high speed communication, and, at the same time, provide *location transparent communication* between servers. In other words, each server should be able to communicate with every other server with similar speed and latency. This requirement ensures that *applications need not be location aware* and, at the same time, it reduces the complexity of the system management.

Typically, the networking infrastructure is organized hierarchically. The servers of a WSC are packed into racks and interconnected by a rack router. Then rack routers are connected to cluster routers which in turn are interconnected by a local communication fabric. Finally, inter data center networks connect multiple WSCs [277]. Clearly, in a hierarchical organization true location transparency is not feasible. Cost considerations ultimately decide the actual organization and performance of the communication fabric.

Oversubscription is a particularly useful measure of the fitness of an interconnection network for a large scale cluster. Oversubscription is defined as the ratio of the worst-case achievable aggregate bandwidth among the servers to the total bisection bandwidth of the interconnect. An oversubscription

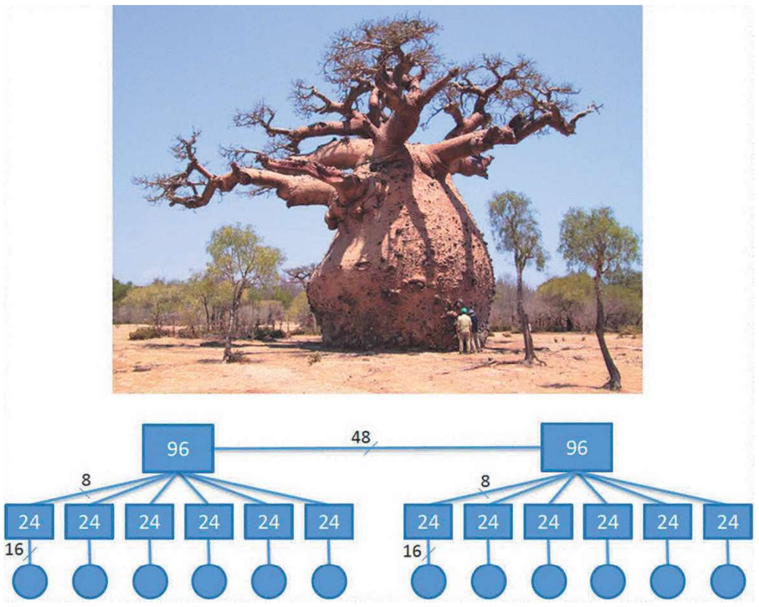


FIGURE 5.10

Fat-trees. (Top) A fat-tree in nature. (Bottom) A 192 node fat-tree interconnection network with two 96-way and twelve 24-way switches in a computer cloud.

of one to one indicates that any host may communicate with an arbitrary hosts at the full bandwidth of the interconnect. An oversubscription value of 4 to 1 means that only 25% of the bandwidth available to servers can be attained for some communication patterns. Typical oversubscription figures are in the 2.5 to 1 and 8 to 1 range.

The cost of the routers and the number of cables interconnecting the routers are major components of the overall cost of an interconnection network. Wire density has scaled up at a slower rate than processor speed and wire delay has remained constant over time thus, better performance and lower costs can only be achieved with innovative router architecture. This motivates the need to take a closer look at the actual design of routers.

Fat-trees. A special instance of the Clos topology discussed in Section 5.7, fat-trees, are optimal interconnects for large-scale clusters and, by extension, for WSCs. When using a fat-tree interconnect servers are placed at the leafs of the tree, while switches populate the root and the internal nodes of the tree. Fat-trees have additional links to increase the bandwidth near the root of the tree. Some set of paths in a fat-tree will saturate all bandwidth available to the end hosts for arbitrary communication patterns. A fat-tree communication architecture can be built with cheap commodity parts as all switching elements of a fat-tree are identical.

Figure 5.10 shows a 192 node fat-tree built with two 96-way switches and twelve 24-way switches. The two 96-way switches at the root are connected via 48 links. Each 24-way switch has 6 × 8 up-

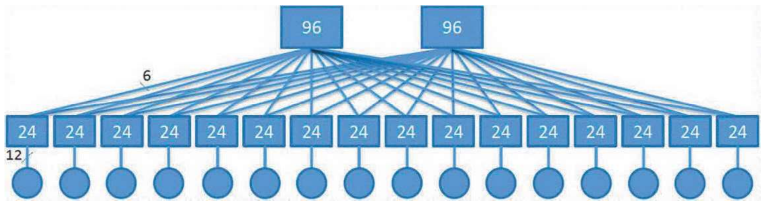


FIGURE 5.11

A 192 node fat-tree interconnect with two 96-way and twelve 24-way switches.

Table 5.2 A side-by-side comparison of performance and cost figures of several interconnection network topologies for 64 nodes.

Property	2D mesh	2D torus	Hypercube	Fat-tree	Fully connected
BW in # of links	8	16	32	32	1024
Max/Avg hop count	14/7	8/4	6/3	11/9	1/1
I/O ports per switch	5	5	7	4	64
Number of switches	64	64	64	192	64
Total number of links	176	192	256	384	2080

link connections to the root and 6 × 16 down connections to 16 servers. Another 192 node fat-tree interconnect with two 96-way and twelve 24-way switches is shown in Figure 5.11.

Table 5.2 from [228] summarizes the performance/cost for a 2D-mesh, a 2D-torus, a Hypercube of order 7, a fat-tree, and a fully connected network. Two dimensions of interconnection network performance, the bisection bandwidth and the average and maximum number of hops, along with three elements affecting the cost, the number of ports per switch, the number of switches, and the total number of links are shown. The fat-tree has the largest bisection bandwidth with the smallest number of I/O ports per switch while the fully connected interconnect has a prohibitively large number of links.

5.7 MULTISTAGE INTERCONNECTION NETWORKS

There are *low-radix* and *high-radix* routers. The former have a small number of ports, while the latter have a large number of ports. The number of intermediate routers in a high-radix network is greatly reduced. Such networks enjoy lower latency and reduced power consumption.

Every five years during the past two decades the pin bandwidth of the chips used for switching has increased by approximately an order of magnitude as a result of the increase in the signaling rate and in the number of signals. High-radix chips divide the bandwidth into a larger number of narrow ports while low-radix chips divide the bandwidth into a smaller number of wide ports.

Clos networks. Clos networks were invented in early 1950s by Charles Clos from Bell Labs [112]. A Clos network is a multistage non-blocking network with an odd number of stages, see Figure 5.12A.

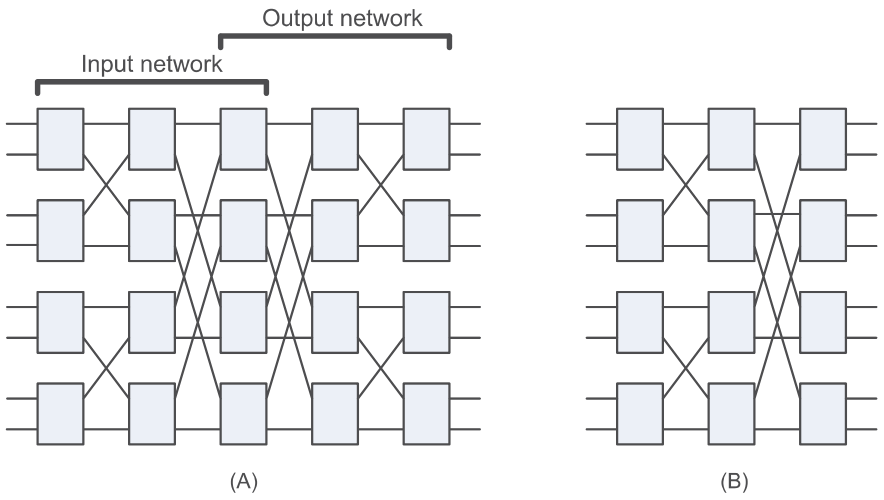


FIGURE 5.12

(A) A 5-stage Clos network with radix-2 routers and unidirectional channels; the network is equivalent to two back-to-back butterfly networks. (B) The corresponding folded-Clos network with bidirectional channels; the input and the output networks share switch modules.

The network consists of two butterfly networks where the last stage of the input is fused with the first stage of the output.

The name butterfly comes from the pattern of inverted triangles created by the interconnections, which look like butterfly wings. A butterfly network transfers the data using the most efficient route, but it is blocking, it cannot handle a conflict between two packets attempting to reach the same port at the same time. In a Clos network all packets overshoot their destination and then hop back to it. Most of the time the overshoot is not necessary and increases the latency, a packet takes twice as many hops as it really needs.

In a *folded Clos* topology the input and the output networks share switch modules. Such networks are sometimes called *fat-tree*; many commercial high-performance interconnects such as Myrinet, InfiniBand, and Quadrics implement a fat-tree topology. Some folded Clos networks use low-radix routers, e.g., the Cray XD1 uses radix-24 routers. The latency and the cost of the network can be lowered using high-radix routers.

The *Black Widow* topology extends the folded Clos topology and has a lower cost and latency; it adds side links and this permits a statical partitioning of the global bandwidth among peer subtrees [447]. The Black Widow topology is used in Cray computers.

Flattened butterfly networks. The flattened butterfly topology [270] is similar to the generalized hypercube that was proposed in the early 1980s, but the wiring complexity is reduced and this topology is able to exploit high-radix routers. When constructing a flattened butterfly we start with a conventional butterfly and combine the switches in each row into a single, higher-radix one. Each router is linked to more processors and this halves the number of router-to-router connections.

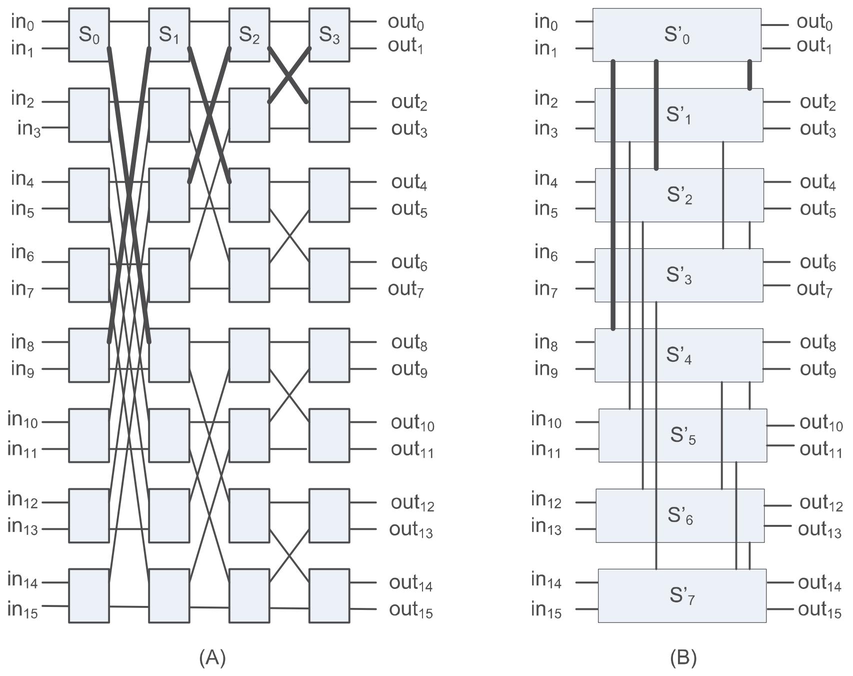


FIGURE 5.13

(A) A 2-ary 4-fly butterfly with unidirectional links. (B) The corresponding 2-ary 4-flat flattened butterfly is obtained by combining the four switches S_0, S_1, S_2 and S_3 in the first row of the traditional butterfly into a single switch S_0' and by adding additional connections between switches [270].

The latency is reduced as data from one processor can reach another processor with fewer hops, though the physical path may be longer. For example, in Figure 5.13A we see a 2-ary 4-fly butterfly; we combine the four switches S_0, S_1, S_2 and S_3 in the first row into a single switch S_0'. The flattened butterfly adaptively senses congestion and overshoots only when it needs to. On adversarial traffic pattern, the flattened butterfly has a similar performance as the folded Clos, but provides over an order of magnitude increase in performance compared to the conventional butterfly.

The network cost for computer clusters can be reduced by a factor of two when high-radix routers (radix-64 or higher) and the flattened butterfly topology are used according to [271]. The flattened butterfly does not reduce the number of local cables, e.g., backplane wires from the processors to routers, but it reduces the number of global cables. The cost of the cables represents as much as 80% of the total network cost, e.g., for a 4K system the cost savings of the flattened buttery exceed 50%.

5.8 INFINIBAND AND MYRINET

InfiniBand is an interconnection network used by high-performance computing systems, as well as computer clouds. As of 2014 InfiniBand was the most commonly used interconnect in supercomput-

Table 5.3 The evolution of the speed of several high-speed interconnects including SDR, DDR, QDR, FDR, and EDR data rates of InfiniBand.

Network	Year	Speed
Gigabit Ethernet (GE)	1995	1 Gbps
10-GE	2001	10 Gbps
40-GE	2010	40 Gbps
Myrinet	1993	1 Gbps
Fiber Channel	1994	1 Gbps
InfiniBand (IB)	2001	2 Gbps (1X SDR)
	2003	8 Gbps (4X SDR)
	2005	16 Gbps (4X DDR) & 24 Gbps (12X SDR)
	2007	32 Gbps (4X QDR)
	2011	56 Gbps (4X FDR)
	2012	100 Gbps (4X EDR)

ers. The architecture of InfiniBand is based on a switched fabric rather than a shared communication channel. In a shared channel architecture all devices share the same bandwidth; the higher the number of devices connected to the channel, the less bandwidth is available to each one of them.

InfiniBand has a very high throughput and very low latency and it is backed by top companies in the industry, including Dell, HP, IBM, Intel, and Microsoft. Intel manufactures InfiniBand host bus adapters and network switches. From Section 5.7 we know that a switched fabric is fault-tolerant and scalable. Every link of the fabric has exactly one device connected at each end of the link thus, the worst case is the same as the typical case. It follows that the performance of InfiniBand can be much greater than that of a shared communication channel such as the Ethernet.

InfiniBand architecture implements the "Bandwidth-out-of-the-box" concept and delivers bandwidth traditionally trapped inside a server across the interconnect fabric. InfiniBand supports several signaling rates and the energy consumption depends on the throughput. Links can be bonded together for additional throughput as shown in Table 5.3. The architectural specifications of InfiniBand define multiple operational data rates: single data rate (SDR), double data rate (DDR), quad data rate (QDR), fourteen data rate (FDR), and enhanced data rated (EDR).

The signaling rates are: 2.5 Gbps in each direction per connection for an SDR connection; 5 Gbps for DDR; 10 Gbps for QDR; 14.0625 Gbps for FDR; 25.78125 Gbps for EDR per lane. The SDR, DDR and QDR link encoding is 8 B/10 B, every 10 bits sent carry 8 bits of data. Thus single, double, and quad data rates carry 2, 4, or 8 Gbit/s useful data, respectively, as the effective data transmission rate is four-fifths the raw rate.

InfiniBand allows links to be configured for a specified speed and width; the reactivation time of the link can vary from several nanoseconds to several microseconds. Exadata and Exalogic systems from Oracle implement the InfiniBand QDR with 40 Gbit/s (32 Gbit/s effective) using Sun Switches. The InififiniBand fabric is used to connect compute nodes, compute nodes with storage servers, and Exadata and Exalogic systems.

In addition to high throughput and low latency InfiniBand supports quality of service guarantees and failover – the capability to switch to a redundant or standby system. It offers point-to-point bidirectional serial links intended for the connection of processors with high-speed peripherals, such as disks, as well as multicast operations.

Note also that the deployment of InfiniBand pushes the limitations of the buses used in personal computers (PCs) and other personal devices as well. Introduced as a standard PC architecture in early 90's the PCI bus has evolved from 32 bit/33 MHz to 64 bit/66 MHz while PCH-X has doubled the clock rate to 133 MHz. Buses have severe electrical, mechanical, and power issues. The number of pins of a parallel bus is quite large; the number of pins necessary for each connection is quite large, e.g., a 64 bit PCI bus requires 90 pins.

Myrinet is an interconnect for massively parallel systems developed at Caltech and later at a company called Myricom. Its main features are [69]:

1. Robustness ensured by communication channels with flow control, packet framing, and error control.
2. Self-initializing, low-latency, cut-through switches.
3. Host interfaces that can map the network, select routes, and translate from network addresses to routes, as well as handle packet traffic.
4. Streamlined host software that allows direct communication between user processes and the network.

The design of Myrinet benefits from the experience gathered by a project to construct a high-speed local-area network at USC/ISI. A Myrinet is composed of point-to-point, full-duplex links connecting hosts and switches, an architecture similar to the one of the ATOMIC project at USC. Multiple-port switches may be connected to other switches and to the single-port host interfaces in any topology. Transmission is synchronous at the sending end, while the receiver circuits are asynchronous. The receiver injects control symbols in the reverse channel of the link for flow control. Myrinet switches use blocking-cut-through routing similar to the one in Intel Paragon and Cray T3D.

Myrinet supports high data rates; a Myrinet link consists of a full-duplex pair of 640 Mbps channels and has regular topology with elementary routing circuits in each node. The network is scalable, its aggregate capacity grows with the number of nodes. Simple algorithmic routing avoids deadlocks and allows multiple packets to flow concurrently through the network.

There is a significant difference between *store and forward* and *cut-through or wormhole* networks. In the former an entire packet is buffered and its checksum is verified in each node along the path from the source to the destination. In wormhole networks the packet is forwarded to its next hop as soon as the header is received and decoded. This decreases the latency, but a packet can still experience blocking if the outgoing channel expected to carry it to the next node is in use. In this case the packet has to wait until the channel becomes free.

A comparison of Myrinet and Ethernet performance as a communication substrate for MPI libraries is presented in [321]. MPI library implementations for Ethernet have a higher message latency and lower message bandwidth because they use the OS network protocol stack. The NAS benchmarks,[6] MG messaging over Myrinet only achieves a 5% higher performance then TCP messaging over Ethernet.

[6]NASA Parallel Benchmarks, NAS, are used to evaluate the performance of parallel supercomputers. The original benchmark included five kernels: IS – Integer Sort, random memory access, EP – Embarrassingly Parallel, CG – Conjugate Gradient, MG – Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive, FT – discrete 3D Fast Fourier Transform, all-to-all communication.

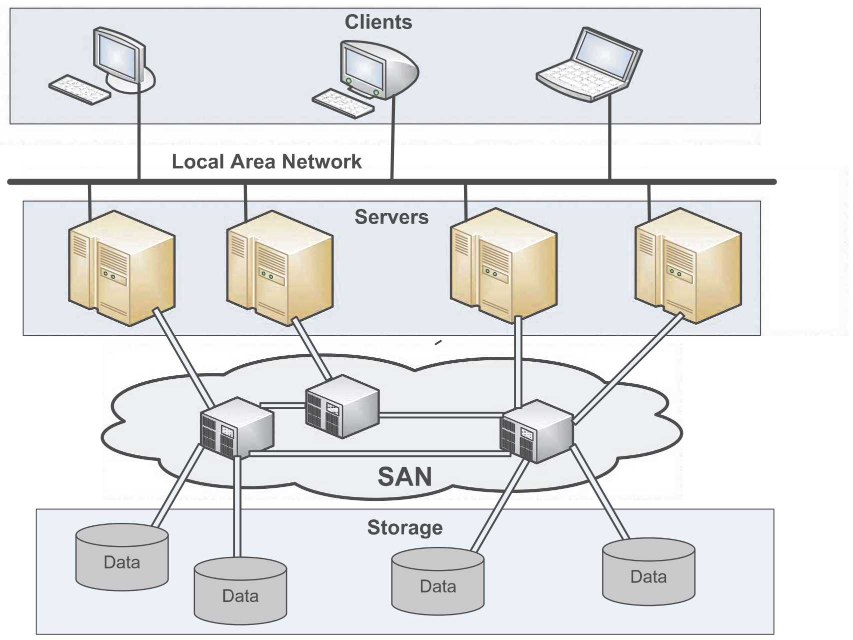


FIGURE 5.14

A storage area network interconnects servers to servers, servers to storage devices, and storage devices to storage devices. Typically it uses fiber optics and the FC protocol.

5.9 STORAGE AREA NETWORKS AND THE FIBRE CHANNEL

A storage area network (SAN) is a specialized, high-speed network for data block transfers. SANs interconnect servers to servers, servers to storage devices, and storage devices to storage devices, see Figure 5.14. A SAN consists of a communication infrastructure and a management layer. The Fibre Channel (FC) is the dominant architecture of SANs.

FC it is a layered protocol with several layers depicted in Figure 5.15:

A. The three lower layer protocols: FC-0, the physical interface; FC-1, the transmission protocol responsible for encoding/decoding; and FC-2, the signaling protocol responsible for framing and flow control. FC-0 uses laser diodes as the optical source and manages the point-to-point fiber connections; when the fiber connection is broken, the ports send a series of pulses until the physical connection is re-established and the necessary handshake procedures are followed.

FC-1 controls the serial transmission and integrates data with clock information. It ensures encoding to the maximum length of the code, maintains DC-balance, and provides word alignment. FC-2 provides the transport methods for data transmitted in 4-byte ordered sets containing data and control characters. It handles the topologies, based on the presence or absence of a fabric, the communication models, the classes of service provided by the fabric and the nodes, sequence and exchange identifiers, and segmentation and reassembly.

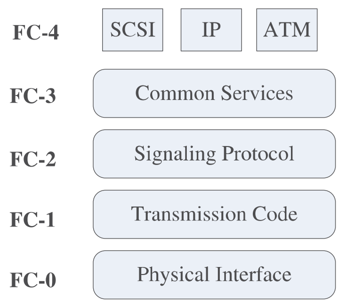


FIGURE 5.15

FC protocol layers. The FC-4 supports communication with the Small Computer System Interface (SCSI), the IP, and the Asynchronous Transfer Mode (ATM) network interfaces.

B. Two upper layer protocols: FC-3, the common services layer; and FC-4, the protocol mapping layer. FC-3 supports multiple ports on a single-node or fabric using:

- Hunt groups – sets of associated ports assigned an alias identifier that allows any frame containing that alias to be routed to any available port within the set.
- Striping to multiply bandwidth, using multiple ports in parallel to transmit a single information unit across multiple links.
- Multicast and broadcast to deliver a single transmission to multiple destination ports or to all ports.

To accommodate different application needs, FC supports several classes of service:

Class 1: rarely used blocking connection-oriented service; acknowledgments ensure that the frames are received in the same order in which they are sent, and reserve full bandwidth for the connection between the two devices.

Class 2: acknowledgments ensure that the frames are received; allows the fabric to multiplex several messages on a frame-by-frame basis; as frames can take different routes it does not guarantee in-order delivery, it relies on upper layer protocols to take care of frame sequence.

Class 3: datagram connection; no acknowledgments.

Class 4: connection-oriented service. Virtual circuits (VCs) established between ports, guarantee in-order delivery and acknowledgment of delivered frames. The fabric is responsible for multiplexing frames of different VCs. Supports Guaranteed Quality of Service (QoS), including bandwidth and latency. This layer is intended for multimedia applications.

Class 5: isochronous service for applications requiring immediate delivery, without buffering.

Class 6: supports dedicated connections for a reliable multicast.

Class 7: similar with Class 2, but used for the control and management of the fabric; a connection-less service with notification of non-delivery.

While every device connected to a LAN has a unique MAC address, each FC device has a unique ID called the WWN (World Wide Name), a 64 bit address. Every port in the switched fabric has its

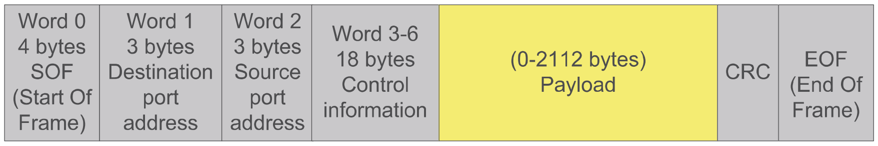


FIGURE 5.16

The format of an FC frame; the payload can be at most 2112 bytes, larger data units are carried by multiple frames.

own unique 24-bit address consisting of: the domain (bits 23–16), the area (bits 15–08), and the port physical address (bits 07–00).

The switch of a switched fabric environment assigns dynamically and maintains the port addresses. When a device with a WWN address logs into a given port, the switch maintains the correlation between that port address and the WWN address of the device using the Name Server. The Name Server is a component of the fabric operating system, which runs inside the switch.

The format of an FC frame is shown in Figure 5.16. Zoning permits finer segmentation of the switched fabric; only the members of the same zone can communicate within that zone. It can be used to separate different environments, e.g., a Microsoft Windows NT from a UNIX environment.

Several other protocols are used for SANs. Fibre Channel over IP (FCIP) allows transmission of Fibre Channel information through the IP network using tunneling. Tunneling is a technique for network protocols to encapsulate a different payload protocol; it allows a network protocol to carry a payload over an incompatible delivery-network, or to provide a secure path through an untrusted network.

Tunneling allows a protocol normally blocked by a firewall to cross it wrapped inside a protocol that the firewall does not block. For example, an HTTP tunnel can be used for communication from network locations with restricted connectivity, e.g., behind NATs, firewalls, or proxy servers. Restricted connectivity is a commonly-used method to lock down a network to secure it against internal and external threats.

Internet Fibre Channel Protocol (iFCP) is a gateway-to-gateway protocol that supports communication among FC storage devices in a SAN, or on the Internet using TCP/IP; iFCP replaces the lower-layer Fibre Channel transport with TCP/IP and Gigabit Ethernet. With iFCP, Fibre Channel devices connect to an iFCP gateway or switch and each Fibre Channel session is terminated at the local gateway and converted to a TCP/IP session via iFCP.

5.10 SCALABLE DATA CENTER COMMUNICATION ARCHITECTURES

The question addressed now is: How to organize the communication infrastructure of large cloud data centers to get the best performance at the lowest cost? Several architectural styles for *data center networks* (DCNs) attempting to provide an answer to this question face major challenges:

- The aggregate cluster bandwidth scales poorly with the cluster size.
- The bandwidth needed by many cloud applications comes at a high price and the cost of the interconnect increases dramatically with the cluster size.

Table 5.4 A side-by-side comparison of performance and cost of hierarchical and fat-tree networks over a span of six years [17].

	Hierarchical Network			**Fat-tree**		
Year	**10 GigE**	**Servers**	**Cost/GigE**	**10 GigE**	**Servers**	**Cost/GigE**
2001	28-port	4480	$25.3 K	28-port	5488	$4.5K
2004	32-port	7680	$4.4 K	48-port	27,688	$1.6K
2006	64-port	10,240	$2.1 K	48-port	27,688	$1.2K
2008	128-port	20,480	$1.8 K	48-port	27,688	$0.3K

- The communication bandwidth of DCNs may become oversubscribed by a significant factor depending on the communication patterns.

We only mention two DCN architectural styles, the *three-tier* and the *fat-tree* DCNs. The former has a multiple-rooted tree topology with three layers, core, aggregate, and access. The servers are directly connected to switches at the leafs of the tree at the *access layer*.

Enterprise switches at the root of the tree form the *core layer* and connect together the switches at the *aggregate layer* and also connect the data center to the Internet. The uplinks of the *aggregate layer* switches connect them to the core layer and their download links connect to the access layer. The three-tier DCN architecture is not suitable for computer clouds, it is not particularly scalable, the bisection bandwidth is not optimal, and the switches at the core layer are expensive and power-hungry.

As noted in Section 5.6 the fat-tree topology is optimal for computer clouds, the bandwidth is not severely affected for messages crossing multiple switches and the interconnection network can be built with commodity rather than enterprise switches. An implementation of the fat-tree topology proposed in [17] is discussed in this section. Several principles guide the design of this network:

- The network should scale to a very large number of nodes.
- The fat-tree should have multiple core switches.
- The network should support multi-path routing. The equal-cost multi-path (ECMP) routing algorithm [241] which performs static load splitting among flows should be used.
- The building blocks of the network should be switches with optimal cost/performance ratios.

Table 5.4 shows the performance and the cost of hierarchical and fat-tree interconnection networks expressed in GigE[7] evolved in the span of six years. The cost per GigE of both types of networks decreased by one order of magnitude, yet this cost/performance indicator for fat-tree built with commodity switches is almost an order of magnitude lower than that of the hierarchic networks. The choice of multi-rooted fat-tree topology and multi-path routing is justified because in 2008 the largest cluster that could be supported with a single rooted core 128-port router with 1:1 oversubscription would have been limited to 1280 nodes.

[7]The IEEE 802.3-2008 standard defines GigE as a technology for transmitting Ethernet frames. 1 GigE corresponds to a rate of one gigabit per second, i.e., 10^9 bits per second.

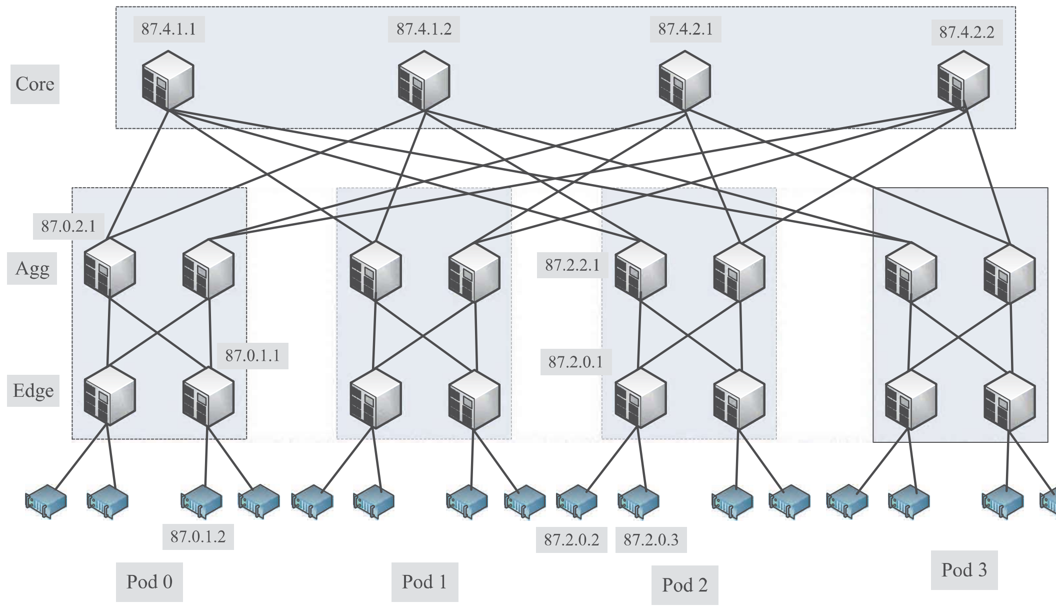


FIGURE 5.17

A fat-tree network with $k = 4$-port switches. Four core 4-way switches are at the root, there are four pods, and two switches at the aggregation layer and two at the edge layer of each pod. Each switch at the edge of a pod is connected to two servers.

A WSC interconnect can be organized as a fat-tree with k-port switches and $k = 48$, but the same fat-tree organization can be supported for any k. The network consists of k *pods*[8] each pod has two layers and $k/2$ switches at each layer. Each switch at the lower layer is connected directly to $k/2$ servers. The other $k/2$ ports are connected to $k/2$ of the k ports in the aggregation layer. The total number of switches is $k(k + 1)$ and the total number of servers connected to the system is k^2. There are $(k/2)^2$ paths connecting every pair of servers.

A WSC with 16 384 servers can be built with 128-port switches and one with 262 144 servers will requires 512-port switches. Figure 5.17 shows a fat-tree interconnection network for $k = 4$. The core, the aggregation, and the edge layers are populated with 4-port switches. Each core switch is connected with one of the switches at the aggregation layer of each pod. The network has four pods, there are four switches at each pod, two at aggregation layer and two at the edge. Four servers are connected to each pod.

The IP addresses of switches have the form $87.pod.switch.1$, and the switches are numbered left to right, and bottom to top. The core switches have addresses of the form $10.k.j.i$ where j and i denote the coordinates of the switch in the $(k/2)^2$ core switch grid starting from top-left. For example, the four switches of pod 2 have IP addresses 87.2.0.1, 87.2.1.1, 87.2.2.1, and 87.2.3.1.

[8]A pod is a repeatable design pattern to maximize the modularity, scalability, and manageability of data center networks.

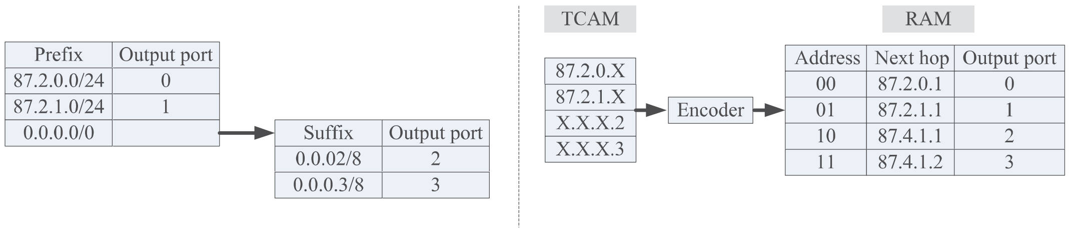


FIGURE 5.18

(Left) The two level routing tables for switch 87.2.2.1. Two incoming packets for IP addresses 10.2.1.2 and 10.3.0.3 are forwarded on ports 1 and 3, respectively. (Right) The RAM implementation of a two-level TCAM routing table.

Servers have IP addresses of the form $87.pod.switch.serverID$ where $serverID$ is the server position in the subnet of the edge router starting from left to right. For example, the IP addresses of the two servers connected to the switch with IP address 87.2.0.1 are 87.2.0.2 and 87.2.0.3.

We can see that there are multiple paths between any pairs of servers. For example, packets sent by the server with IP address 87.0.1.2 to server with IP address 87.2.0.3 can follow the following routes:

$$\begin{array}{l} 87.0.1.1 \mapsto 87.0.2.1 \mapsto 87.4.1.1 \mapsto 87.2.2.1 \mapsto 87.2.0.1 \\ 87.0.1.1 \mapsto 87.0.2.1 \mapsto 87.4.1.2 \mapsto 87.2.2.1 \mapsto 87.2.0.1 \\ 87.0.1.1 \mapsto 87.0.1.1 \mapsto 87.4.2.1 \mapsto 87.2.2.1 \mapsto 87.2.0.1 \\ 87.0.1.1 \mapsto 87.0.1.1 \mapsto 87.4.2.2 \mapsto 87.2.2.2 \mapsto 87.2.0.1 \end{array} \tag{5.5}$$

Packet routing supports two-level prefix lookup and it is implemented using two-level routing tables. This strategy may increase the lookup latency but the prefix search can be done in parallel and could compensate for the latency increase. The main table entries are of the form (*prefix*, *output port*) and could have an additional pointer to a secondary table or could be terminating if none of its entries point to a secondary table. A secondary table consists of (*suffix*, *output port*) entries and may be pointed to by more than one first level entry.

Figure 5.18 (Left) shows the two level routing tables for switch 87.2.2.1 and routing for two incoming packets for servers with the IP addresses 87.2.1.2 and 87.3.0.3; the incoming packets are forwarded on ports 1 and 3, respectively. Lookup engines use a ternary version of content-addressable memory (CAM), called TCAM. Figure 5.18 (Right) shows that TCAM stores address prefixes and suffixes, which index a RAM that stores the IP address of the next hop and the output port.

The prefix entries are stored with numerically smaller addresses first and the right-handed (suffix) entries in larger addresses. The output of the CAM is encoded so that the entry with the numerically smallest matching address is the output. When the destination IP address of a packet matches both a left-handed and a right-handed entry, then the left-handed entry is chosen.

The k switches in a pod have terminating prefixes to the subnets in that pod. When two servers in the same pod but on a different subnets communicate, all upper-level switches of the pod will have a terminating prefix pointing to the destination subnet's switch.

For all outgoing pod traffic, the pod switches have a default /0 prefix with a secondary table matching the least-significant byte of the destination IP address, the server ID. Traffic diffusion occurs only in the first half of a packet's journey. Once a packet reaches a core switch, there is exactly one link to its destination pod, and that switch will include a terminating $/16$ prefix for the pod of that packet $(87.pod.0.0/16, port)$. Once a packet reaches its destination pod, the receiving upper-level pod switch will also include a $(10.pod.switch.0/24, port)$ prefix to direct the packet to its destination subnet switch, where it is finally switched to its destination server.

Each pod switch assigns terminating prefixes for subnets in the same pod and add a /0 prefix with a secondary table matching the $serverID$s for inter-pod traffic. The routing tables for the upper pod switches are generated with the pseudocode of Algorithm 5.1. For the lower pod switches, the /24 subnet prefix in line 3 is omitted since that subnet's own traffic is switched, and intra- and inter-pod traffic should be evenly split among the upper switches.

Algorithm 5.1 Generates aggregation switch routing tables.

```
1  foreach  $pod\ x \in [0, k-1]$  do
2      foreach  $switch\ z \in [k/2, k-1]$  do
3          foreach  $subnet\ i \in [0, k/2-1]$  do
4              $addPrefix(10.x.z.1, 10.x.i.0/24, i);$
5          end
6          $addPrefix(10.x.z.1, 0.0.0.0/0, 0);$
7          foreach  $hostID\ i \in [2, (k/2)+1]$  do
8              $addSuffix(10.x.z.1, 0.0.0.i/8, (i-2+z) \mod (k/2) + (k/2);$
9          end
10     end
11 end
```

Core switches contains only terminating /16 prefixes pointing to their destination pods, as shown in Algorithm 5.2.

Algorithm 5.2 Generates routing tables for core switches.

```
1  foreach  $j \in [0, k-1]$  do
2      foreach  $i \in [1, k/2]$  do
3          foreach  $destination\ pod \in [0, k/2-1]$  do
4              $addPrefix(10.k.j.i, 10.x.0.0/16, x);$
5          end
6      end
7  end
```

The maximum numbers of first-level prefixes and second-level suffixes are k and $k/2$, respectively.

Flow classification with dynamic port-reassignment in pod switches overcomes cases of local congestion when two flows compete for the same output port, even though another port that has the same cost to the destination is underused.

Power consumption and heat dissipation are major concerns for the cloud data centers. Switches at the higher tiers of the interconnect in data centers consume several kW and the entire interconnection infrastructure consumes hundreds to thousands of kW.

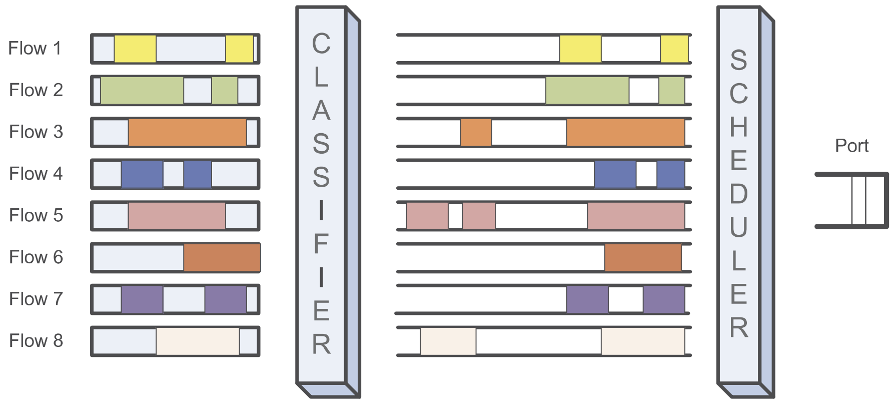


FIGURE 5.19

Fair Queuing – packets are first classified into flows by the system and then assigned to a queue dedicated to the flow; queues are serviced one packet at a time in round-robin order and empty queues are skipped.

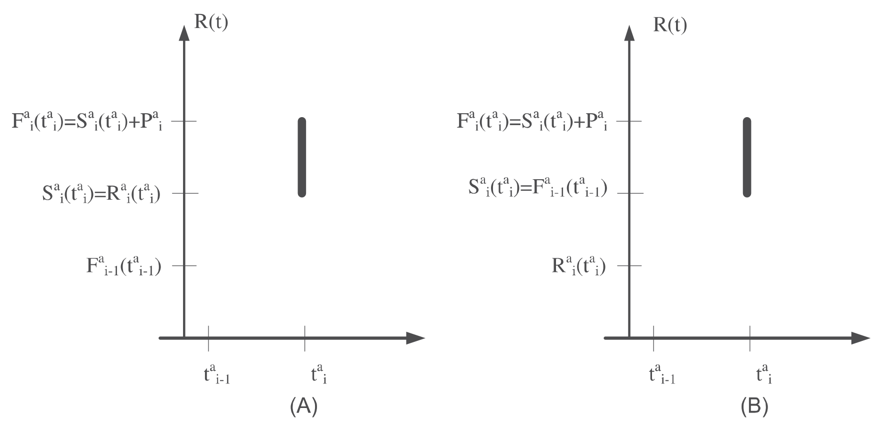
5.11 NETWORK RESOURCE MANAGEMENT ALGORITHMS

A critical aspect of resource management in cloud computing is to guarantee the communication bandwidth required by an application as specified by an SLA. The solutions to this problem are based on the strategies used for some time on Internet to support the data streaming QoS requirements.

Cloud interconnects consist of communication links of limited bandwidth and switches of limited capacity. When the load exceeds its capacity, a switch starts dropping packets because it has limited input buffers for the switching fabric and for the outgoing links, as well as limited CPU cycles. A switch must handle multiple flows, pairs of source-destination end-points of the traffic, thus, a scheduling algorithm has to manage several quantities at the same time: the *bandwidth*, the amount of data each flow is allowed to transport, the *timing* when the packets of individual flows are transmitted, and the *buffer space* allocated to each flow.

Communication and computing require scheduling therefore, it should be no surprise that the first algorithm we discuss can be used for scheduling packets transmission, as well as threads. A first strategy to avoid network congestion is to use a FCFS scheduling algorithm. The advantage of FCFS algorithm is a simple management of the three quantities: bandwidth, timing, and buffer space. Nevertheless, the FCFS algorithm does not guarantee fairness; greedy flow sources can transmit at a higher rate and benefit from a larger share of the bandwidth.

Fair Queuing (FQ). The algorithm ensures that a high-data-rate flow cannot use more than its fair share of the link capacity. Packets are first classified into flows by the system and then assigned to a queue dedicated to the flow. Packet queues are serviced one packet at a time in round-robin (RR) order, Figure 5.19. FQ's objective is *max–min* fairness. This means that first it maximizes the minimum data rate of any of the data flows and then it maximizes the second minimum data rate. Starvation of expensive flows is avoided but the throughput is low.



R(t)
$F^a_i(t^a_i)=S^a_i(t^a_i)+P^a_i$
$S^a_i(t^a_i)=R^a_i(t^a_i)$
$F^a_{i-1}(t^a_{i-1})$
t^a_{i-1}
t^a_i
(A)
R(t)
$F^a_i(t^a_i)=S^a_i(t^a_i)+P^a_i$
$S^a_i(t^a_i)=F^a_{i-1}(t^a_{i-1})$
$R^a_i(t^a_i)$
t^a_{i-1}
t^a_i
(B)

FIGURE 5.20

Transmission of a packet i of flow a arriving at time t_i^a of size P_i^a bits. The transmission starts at time $S_i^a = \max[F_{i-1}^a, R(t_i^a)]$ and ends at time $F_i^a = S_i^a + P_i^a$ with $R(t)$ the number of rounds of the algorithm. (A) The case $F_{i-1}^a < R(t_i^a)$. (B) The case $F_{i-1}^a \geq R(t_i^a)$.

The FQ algorithm guarantees the fairness of buffer space management, but does not guarantee fairness of bandwidth allocation; indeed, a flow transporting large packets will benefit from a larger bandwidth [355].

The FQ algorithm in [139] proposes a solution to this problem. First, it introduces a *Bit-by-bit Round-robin (BR)* strategy; as the name implies, in this rather impractical scheme a single bit from each queue is transmitted and the queues are visited in a round-robin fashion. Let $R(t)$ be the number of rounds of the BR algorithm up to time t and $N_{active}(t)$ the number of active flows through the switch. Call t_i^a the time when the packet i of flow a, of size P_i^a bits arrives and call S_i^a and F_i^a the values of $R(t)$ when the first and the last bit, respectively, of the packet i of flow a are transmitted. Then,

$$F_i^a = S_i^a + P_i^a \quad \text{and} \quad S_i^a = \max[F_{i-1}^a, R(t_i^a)]. \tag{5.6}$$

The quantities $R(t)$, S_i^a and F_i^a in Figure 5.20 depend only on the arrival time of the packets, t_i^a, and not on their transmission time, provided that a flow a is active as long as

$$R(t) \leq F_i^a \quad \text{when} \quad i = \max(j | t_i^a \leq t). \tag{5.7}$$

The authors of [139] use for packet-by-packet transmission time the following non-preemptive scheduling rule which emulates the BR strategy: *the next packet to be transmitted is the one with the smallest* F_i^a. A preemptive version of the algorithm requires that the transmission of the current packet be interrupted as soon as one with a shorter finishing time, F_i^a, arrives.

A fair allocation of the bandwidth does not have an effect on the timing of the transmission. A possible strategy is to allow less delay for the flows using less than their fair share of the bandwidth. The same paper [139] proposes the introduction of a quantity called the *bid*, B_i^a, and scheduling the packet

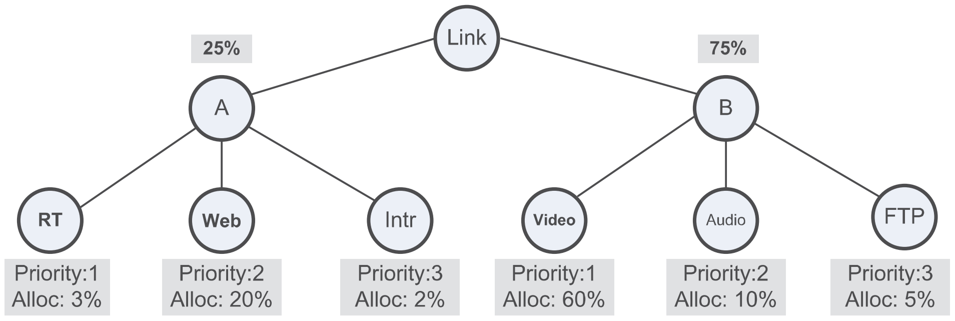


FIGURE 5.21

CBQ link sharing for two groups A, of short-lived traffic, and B, of long-lived traffic, allocated 25% and 75% of the link capacity, respectively. There are six classes of traffic with priorities 1, 2, and 3. The RT (real-time) and the video streaming have priority 1 and are allocated 3% and 60%, respectively, of the link capacity. Web transactions and audio streaming have priority 2 and are allocated 20% and 10%, respectively of the link capacity. Intr (interactive applications) and FTP (file transfer protocols) have priority 3 and are allocated 2% and 5%, respectively, of the link capacity.

transmission based on its value. The bid is defined as

$$B_i^a = P_i^a + \max[F_{i-1}^a, (R(t_i^a) - \delta)], \tag{5.8}$$

with δ a non-negative parameter. The properties of the FQ algorithm, as well as the implementation of a non-preemptive version of the algorithm are analyzed in [139].

The Stochastic Fairness Queuing (SFQ) algorithm is a simpler and less accurate implementation of the FQ algorithms and requires less calculations. SFQ ensures that each flow has the opportunity to transmit an equal amount of data and takes into account data packet sizes [338].

Class-Based Queuing (CBQ). This algorithm is a widely used strategy for link sharing proposed by Sally Floyd and Van Jacobson in 1995 [176]. The objective of CBQ is to support flexible link sharing for applications which require bandwidth guarantees such as VoIP, video-streaming, and audio-streaming. At the same time, CBQ supports some balance between short-lived network flows, such as web searches, and long-lived ones, such as video-streaming or file transfers.

CBQ aggregates the connections and constructs a hierarchy of classes with different priorities and throughput allocations. To accomplish link sharing, CBQ uses several functional units: (i) a *classifier* which uses the information in the packet header to assign arriving packets to classes; (ii) an *estimator* of the short-term bandwidth for the class; (iii) a *selector*, or scheduler, to identify the highest priority class to send next and, if multiple classes have the same priority, to schedule them on a round-robin base; and (iv) a *delayer* to compute the next time when a class that has exceeded its link allocation is allowed to send.

The classes are organized in a tree-like hierarchy; for example, in Figure 5.21 we see two types of traffic, group A corresponding to short-lived traffic and group B corresponding to long-lived traffic. The leaves of the tree are considered Level 1 and in this example include six classes of traffic: real-time,

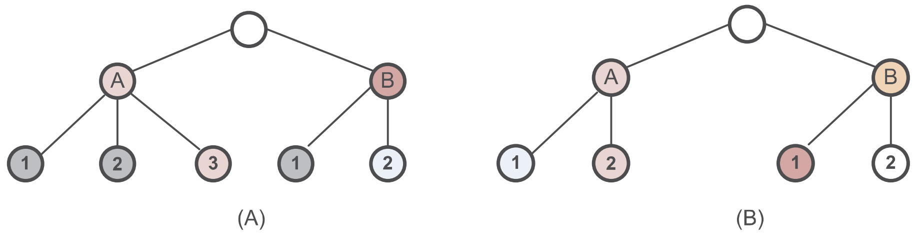


FIGURE 5.22

There are two groups A and B and three types of traffic, e.g., web, real-time, and interactive, denoted as $1, 2,$ and 3. (A) Group A and class $A.3$ traffic are underlimit and unsatisfied; classes $A.1$, $A.2$ and $B.1$ are overlimit, unsatisfied and with persistent backlog and have to be regulated; type $A.3$ is underlimit and unsatisfied; group B is overlimit. (B) Group A is underlimit and unsatisfied; Group B is overlimit and needs to be regulated; class $A.1$ traffic is underlimit; class $A.2$ is overlimit and with persistent backlog; class $B.1$ traffic is overlimit and with persistent backlog and needs to be regulated.

web, interactive, video streaming, audio streaming, and file transfer. At Level 2 there are the two classes of traffic, A and B. The root, at Level 3 is the link itself.

The link sharing policy aims to ensure that if sufficient demand exists then, after some time intervals, each interior or leaf class receives its allocated bandwidth. The distribution of the "excess" bandwidth follows a set of guidelines, but does not support mechanisms for congestion avoidance.

A class is *overlimit* if over a certain recent period it has used more than its bandwidth allocation (in bytes per second), *underlimit* if it has used less, and *atlimit* if it has used exactly its allocation. A leaf class is *satisfied* if it is underlimit and has a persistent backlog and it is *unsatisfied* otherwise; a non-leaf class is unsatisfied if it is underlimit and has some descendent class with a persistent backlog. A precise definition of the term "persistent backlog" is part of a local policy. A class does not need to be *regulated* if it is underlimit or if there are no unsatisfied classes; the class should be regulated if it is overlimit and if some other class is unsatisfied and this regulation should continue until the class is no longer overlimit or until there are no unsatisfied classes, see Figure 5.22 for two examples.

The Linux kernel implements a link sharing algorithm called *Hierarchical Token Buckets* (HTB) inspired by CBQ. In CBQ every class has an *assured rate* (AR); in addition to the AR every class in HTB has also a *ceil rate* (CR), see Figure 5.23. The main advantage of HTB over CBQ is that it allows *borrowing*. If a class C needs a rate above its AR it tries to borrow from its parent; then the parent examines its children and, if there are classes running at a rate lower that their AR, the parent can borrow from them and reallocate it to class C.

5.12 CONTENT DELIVERY NETWORKS

Computer clouds support not only network-centric computing but also network-centric content. For example, we shall see in Chapter 6 that Internet video was expected to generate in 2013 over 18 Exabytes of data per month. Video traffic will account for 79% of the global Internet traffic by 2020. The vast amount of data stored on the cloud has to be delivered efficiently to a large user population.

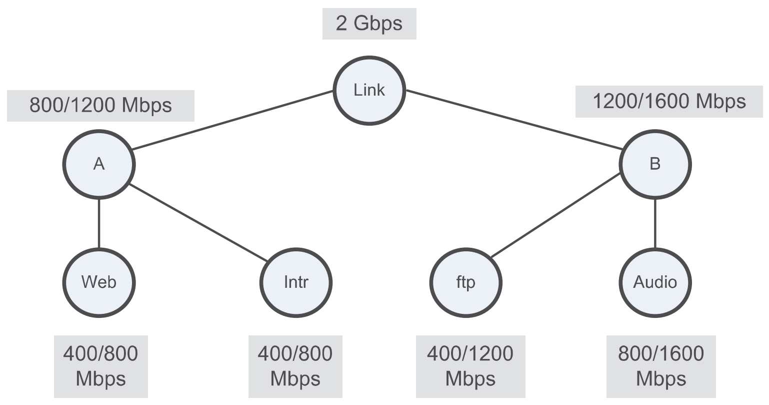


FIGURE 5.23

HTB packet scheduling uses for every node a ceil rate in addition to the allowed rate.

Content Delivery Networks (CDNs) offer fast and reliable content delivery and reduce communication bandwidth by caching and replication. A CDN receives the content from an *Origin* server, then replicates it to its *Edge* cache servers; the content is delivered to an end-user from the "closest" Edge server.

CDNs are designed to support scalability, to increase reliability and performance, and to provide better security. The volume of transactions and data transported by the Internet increases dramatically every year; additional resources are necessary to accommodate the additional load placed on the communication and storage systems and to improve the end-user experience. CDNs place additional resources provisioned to absorb the traffic caused by *flash crowds*[9] and, in general, to provide capacity on demand.

The additional resources are placed strategically throughout the Internet to ensure scalability. The resources provided by a CDN are replicated and when one of the replicas fails, the content is available from another one; the replicas are "close" to the consumers of the content and this placement reduces the start-up time and the communication bandwidth. A CDN uses two types of servers; the *origin* server updated by the content provider and *replica* servers which cache the content and serve as authoritative reference for client requests. Security is a critical aspect of the services provided by a CDN; the replicated content should be protected from the increased risk of cyber fraud and unauthorized access.

A CDN can deliver static content and/or live or on-demand streaming media. *Static content* refers to media that can be maintained using traditional caching technologies because changes are infrequent; examples of static content are: HTML pages, images, documents, software patches, and audio and/or video files. *Live media* refers to live events when the content is delivered in real time from the encoder to the media server. On-demand delivery of audio and/or video streams, movie files and music clips

[9]The term flash crowds refers to an event which disrupts the life of a very significant segment of the population, such as an earthquake in a very populated area, and causes the Internet traffic to increase dramatically.

provided to the end-users is content-encoded and then stored on media servers. Virtually all CDN providers support static content delivery, while live or on-demand streaming media is considerably more challenging.

CDN providers and protocols. The first CDN was setup by *Akamai*, a company evolved from an MIT project to optimize network traffic. Since its inception Akamai has placed some 20 000 servers in 1000 networks in 71 countries; in 2009 it controlled some 85% of the market [394].

Akamai mirrors the contents of clients on multiple systems placed strategically through the Internet. Though the domain name (but not sub-domain) is the same, the IP address of the resource requested by a user points to an Akamai server rather than the customer's server. Then the Akamai server is automatically picked depending on the type of content and the network location of the end user.

There are several other active commercial CDNs including EdgeStream providing video streaming and Limelight Networks providing distributed on-demand and live delivery of video, music, and games. There are several academic CDNs: Coral is a freely-available network designed to mirror web content, hosted on PlanetLab; Globule is an open-source collaborative CDN developed at the Vrije Universiteit in Amsterdam.

The communication infrastructure among different CDN components uses a fair number of protocols including: Network Element Control Protocol (NECP), Web Cache Coordination Protocol (WCCP), SOCKS, Cache Array Routing Protocol (CARP), Internet Cache Protocol (ICP), Hypertext Caching Protocol (HTCP), and Cache Digest described succinctly in [394]. For example, caches exchange ICP queries and replays to locate the best sites to retrieve an object; HTCP is used to discover HTTP caches, to cache data, to manage sets of HTTP caches and monitor cache activity.

CDN organization, design decisions, and performance. There are two strategies for CDN organization; in the so-called *overlay*, the network core does not play an active role in the content delivery. On the other hand, the *network* approach requires the routers and the switches to use dedicated software to identify specific application types and to forward user's requests based on predefined policies.

The first strategy is based exclusively on content replication on multiple caches and redirection based on proximity to the end-user. In the second approach the network core elements redirect content requests to local caches or redirect data center's incoming traffic to servers optimized for specific content type access. Some CDNs including Akamai use both strategies.

Important design and policy decisions for a CDN are:

1. Placement of the edge servers.
2. Content selection and delivery.
3. Content management.
4. Request routing policies.

The placement problem is often solved with suboptimal heuristics using as input the workload patterns and the network topology. The simplest, but a costly approach for content selection and delivery, is the *full-site* replication suitable for static content; the edge servers replicate the entire content of the origin server. On the other hand, the *partial-site* selection and delivery retrieves the base HTML page from the origin server and the objects referenced by this page from the edge caches. The objects can be replicated based on their popularity, or on some heuristics.

The content management depends on the caching techniques, the cache maintenance and the cache update policies. CDNs use several strategies to manage the consistency of content at replicas: periodic updates, updates triggered by the content change, and on-demand updates.

The request-routing in a CDN directs users to the closest edge server that can best serve the request; metrics, such as network proximity, client perceived latency, distance, and replica server load are taken into account when routing a request. Round-robin is a non-adaptive request-routing which aims to balance the load; it assumes that all edge servers have similar characteristics and can deliver the content.

Adaptive algorithms perform considerably better, but are more complex and require some knowledge of the current state of the system. The algorithm used by *Akamai* takes into consideration metrics such as: the load of the edge server, the bandwidth currently available to a replica server, the reliability of the connection of the client to the edge servers.

CDN routing can exploit an organization where several edge servers are connected to a service node aware of the load and the information about each edge server connected to it and attempts to implement a *global load balancing policy*. An alternative is *DNS-based routing* when a domain name has multiple IP addresses associated to it and the service provider's DNS server returns the IP addresses of the edge servers holding the replica of the requested object; then the client's DNS server chooses one of them.

Another alternative is the *HTTP redirection*; in this case a web server includes in the HTTP header of a response to a client the address of another edge server. Finally, *IP anycasting* requires that the same IP address is assigned to several hosts and the routing table of a router contains the address of the host closest to it.

The critical metrics of CDN performance are:

1. Cache hit ratio – the ratio of the number of cached objects versus total number of objects requested.
2. Reserved bandwidth for the origin server.
3. Latency – it is based on the perceived response time by the end users.
4. Edge server utilization.
5. Reliability – based on packet loss measurements.

CDNs will face considerable challenges in the future due to increased appeal of data streaming and to the proliferation of mobile devices such as of smart phones and tablets. On-demand video streaming requires enormous bandwidth and storage space, as well as powerful servers; CDNs for mobile networks must be able to dynamically reconfigure the system in response to spatial and temporal demand variations.

Content-Centric Networks. Content-Centric Networks (CCNs) are related to information-centric networking architectures such as Named Data Networks (NDNs) discussed in Section 5.4 where content is named and transferred throughout the network. In such networks the request for a named content is routed to the producer or to any entity that can deliver the expected content object.

CCNs content may be served from the cache of any router. Content is signed by its producer and the consumer is able to verify the signature before actually using the content. CCNs supports opportunistically caching. According to http://chris-wood.github.io/2015/06/16/CCN-vs-CDN.html CCNs offer a number of advantages. The popularity of the content need not be predicted beforehand, while providing the benefits not offered by IP-based CDNs including:

1. Active and intelligent forwarding strategies for routers.
2. Publisher mobility is easily supported via CCN routing protocols.
3. Congestion control can be enforced within the network.
4. The existing (and problematic) IP stack can be completely replaced with a new set of layers.
5. Existing APIs can be completely reworked to focus on content, not on addresses.
6. Content security is not tied to the channel, but to the content itself.

Table 5.5 VANETs applications, contents, local interest, local validity, and lifetime.

Application	Contents	Local interest	Local validity	Lifetime
Safety warnings	Dangerous Road	All	100 m	10 s
Safety warnings	Accident	All	500 m	30 s
Safety warnings	Work zone	All	1 km	Construction
Public service	Emergency vehicle	All	500 m	10 min
Public service	Highway information	All	5 km	All day
Driving	Road congestion	All	5 km	30 min
Driving	Navigation Map	Subscribers	5 km	30 min

The concept of *content service network* (CSN) was introduced in [318]. CSNs are overlay networks built around CDNs to provide an infrastructure service for processing and transcoding.

5.13 VEHICULAR AD HOC NETWORKS

A vehicular ad hoc network (VANET) consists of groups of moving or stationary vehicles connected by a wireless network. Until recently the main use of VANETs was to provide safety and comfort to drivers in vehicular environments. This view is changing, vehicular ad hoc networks are seen now as an infrastructure for an intelligent transportation system with increasing number of autonomous vehicles, and for any activity requiring Internet connectivity in a smart city. Also, VANETs allow on-board computers of mostly stationary vehicles, e.g., vehicles at an airport parking, to serve as resources of a mobile computer cloud with minimum help from the Internet infrastructure.

The contents produced and consumed by vehicles has *local relevance* in terms of time, space, and agents involved, the producer and the consumer. Vehicle-generated information has *local validity*, a limited spatial scope, an *explicit lifetime*, a limited temporal scope, and *local interest*, it is relevant to agents in a limited area around the vehicle. For example, the information that a car is approaching a congested area of a highway is relevant only for that particular segment of the road, at a particular time, and for vehicles nearby. The attributes of vehicular contents are summarized in Table 5.5 from [299].

One of the distinguishing characteristics of VANETs is the *content-centric distribution*, the content is important, the source is not. This is in marked contrast to the Internet where an agent demands information from a specific source. For example, traffic information floods a specific area and vehicle retrieves it without concern for the source of it, while an Internet request for highway traffic information is directed to a specific site. Vehicle applications collect sensor data and vehicles collaborate sharing sensory data. Sensory data is collected by vehicle-installed cameras, by on-board instruments. For example, *CarSpeak* allows a vehicle to access sensors on neighboring vehicles in the same manner in which it can access its own [284]. *Waze* is a community-based traffic and navigation application allowing drivers real-time traffic and road information.

VANET communication protocols are similar to the ones used by wired networks, each host has an IP address. Assigning IP addresses to moving vehicles is far from trivial and often requires a Dynamic Host Configuration Protocol (DHCP) server, a heresy for ad hoc networks that operate without any infrastructure, using self-organization protocols. Vehicles frequently join and leave the network and content of interest cannot be consistently bound to a unique IP address. A router typically relays and then deletes content.

5.14 FURTHER READINGS

The “Brief history of the Internet” [288] was written by the Internet pioneers Barry Leiner, Vinton Cerf, David Clark, Robert Kahn, Leonard Kleinrock, Daniel Lynch, Jon Postel, Larry Roberts, and Stephen Wolff.

The widely used text of Kurose and Ross [283] is an excellent introduction to basic networking concepts. The book by Bertsekas and Gallagher [65] gives insights into the performance evaluation of computer networks. The classic texts on queuing theory of Kleinrock [274] are a required reading for those interested in network analysis.

A survey of information-centric networking research is given in [532] while [548] is a succinct presentation of NDNs. Several publications related to software-defined networks are available on the site of the Open Network Foundation, https://www.opennetworking.org/sdn-resources.

The Moore’s Law for traffic is discussed in [354]. The class-based queuing algorithm was introduced by Floyd and Van Jacobson in [176]. The *Black Widow* topology for system interconnects is analyzed in [447]. An extensive treatment of Storage Area Networks can be found in [481].

Alternative organizations of networks have been discussed in the literature. Scale-free networks and their applications are described by Barabási and Albert in [14–16,51]. The small-worlds networks were introduced by Watts and Strogatz in [515]. Epidemic algorithms for the dissemination of topological information are presented in [210,255,256]. Erdös–Rény random graphs are analyzed in [71,165]. Energy efficient protocols for cooperative networks are discussed in [163].

The future of fiber networks is covered in [289]. Vehicle ad hoc networks and their applications to vehicular cloud computing are discussed in [191,299,518]. An analysis of peer-to-peer networks is reported in [192]. Network management for private clouds, p2p networks, and virtual networks are presented in [351], [352], and [359], respectively.

5.15 EXERCISES AND PROBLEMS

Problem 1. Four ground rules for an open-architecture principle are cited in the “Brief history of the Internet.” Read the paper and analyze the implication of each one of these rules.

Problem 2. The paper in Problem 1 lists also several key issues for the design of the network: (1) algorithms to prevent lost packets from permanently disabling communications and enabling them to be successfully retransmitted from the source; (2) providing for host-to-host “pipelining” so that multiple packets could be en-route from source to destination at the discretion of the participating hosts, if the intermediate networks allowed it; (3) the need for end-end checksums, reassembly of packets from fragments and detection of duplicates, if any; (4) the need for global addressing; and (5) techniques for host-to-host flow control.

Discuss how these issues were addressed by the TCP/IP network architecture.

Problem 3. Analyze the challenges of transition to IPv6. What will be in your view the effect of this transition on cloud computing?

Problem 4. Discuss the algorithms used to compute the TCP window size.

Problem 5. Creating a virtual machine (VM) reduces ultimately to copying a file, therefore the explosion of the number of VMs cannot be prevented, see Section 11.9. As each VM needs its own IP address, virtualization could drastically lead to an exhaustion of the IPv4 address space. Analyze the solution to this potential problem adopted by the IaaS cloud service delivery model.

Problem 6. Read the paper describing the stochastic fair queuing algorithm [176]. Analyze the similarities and dissimilarities of this algorithm and the start-time fair queuing discussed in Section 9.13.

Problem 7. The small-worlds networks were introduced by D. Watts and S. H. Strogatz. They have two desirable features, high clustering and small path length. Read [515] and design an algorithm to construct a small-worlds network.

Problem 8. The properties of scale-free networks are discussed in [14–16,51]. Discuss the important features of systems interconnected by scale-free networks discussed in these papers.

Problem* 9. Consider two 192 node fat-tree interconnect with two 96-way and twelve 24-way switches, the one in Figure 5.10 and the one in Figure 5.11. Compute the bisection bandwidth of the two interconnects.