A firewall can monitor and control access points. In such cases, the network administrator wants to know about any intrusion from a suspicious source to the network. For example, a host in a network can be attacked by receiving a large number of SYN packets.

- *Billing and accounting management.* The network administrator specifies user access or restrictions to network resources and issues all billing and charges, if any, to users.

Locating a failing point, such as an adapter failure at a host or a router, can be done by appropriate network management tools. Normally, a standard packet format is specified for network management.

9.7.1 Elements of Network Management

Network management has three main components: network management: a *managing center*, a *managed device*, and a *network management protocol*. The managing center consists of the network administrator and his or her facilities. Typically, the managing center comprises a substantial human network. A managed device is the network equipment, including its software, that is controlled by the managing center. Any hub, bridge, router, server, printer, or modem can be a managed device. The network management protocol is a policy between the managing center and the managed devices. The protocol in this context allows the managing center to obtain the status of managed devices. In network management, an *agent* is a managed device, such as a router, hub, or bridge. A *manager* is a network administrative device, as a management host. An agent can use the network management protocol to inform the managing center of an unexpected event.

9.7.2 Structure of Management Information (SMI)

The *structure of management information* (SMI) language is used to define the rules for naming objects and to encode objects in a managed network center. In other words, SMI is a language by which a specific instance of the data in a managed network center is defined. For example, Integer32 means a 32-bit integer with a value between -2^{31} and $-2^{31} - 1$. The SMI language also provides higher-level language constructs, which typically specify the data type, status, and semantics of managed objects containing the management data. For example, the STATUS clause specifies whether the object definition is current or obsolete, ipInDelivers defines a 32-bit counter to trace the number of IP datagrams received at a managed device and then received at an upper-layer protocol.

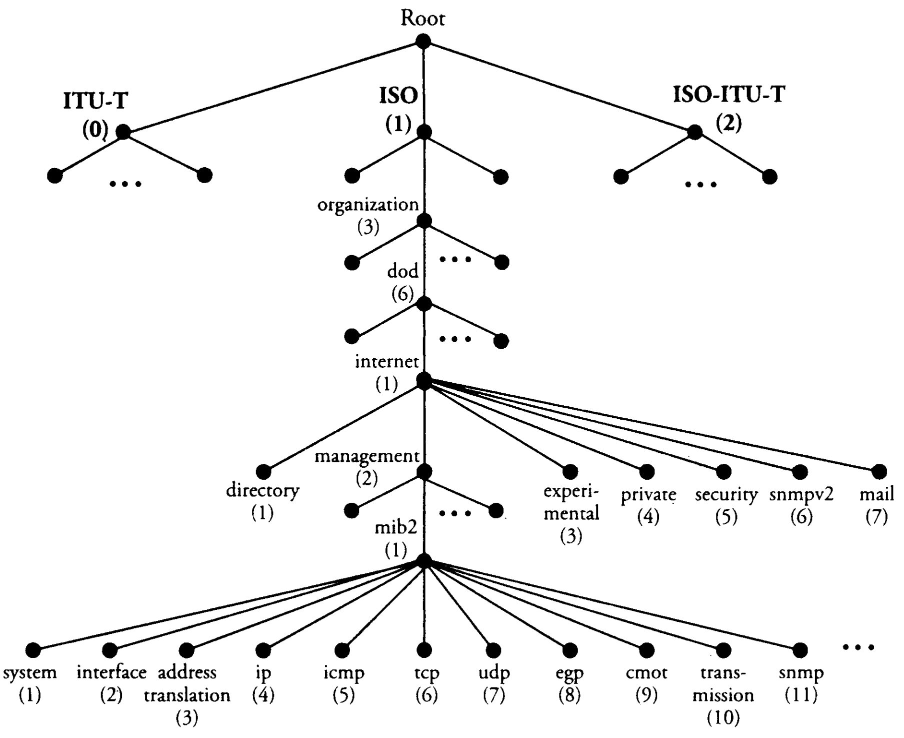


Figure 9.11 ASN.1 object identifier organized hierarchically

9.7.3 Management Information Base (MIB)

Management information base (MIB) is an information storage medium that contains managed objects reflecting the current status of the network. Because managed objects have associated pieces of information that are stored in a MIB, the MIB forms a collection of named objects, including their relationships to one another in a management center. The information pieces can be obtained by directing the managing center to do so.

Objects are organized in a hierarchical manner and are identified by the *abstract syntax notation one* (ASN.1) object definition language. The hierarchy of object names, known as *ASN.1 object identifier*, is an object identifier tree in which each branch has both a name and a number, as shown in Figure 9.11. Network management can then identify an object by a sequence of names or numbers from the root to that object.

On the root of the object identifier hierarchy are three entries: ISO (International Standardization Organization), ITU-T (International Telecommunication Union-Telecommunication) standardization sector, and ISO-ITU-T, the joint branch of these two organizations. Figure 9.11 shows only part of the hierarchy. Under the ISO entry are other branches. For example, the *organization (3)* branch is labeled sequentially from the root as 1.3. If we continue to follow the entries on this branch, we see a path over *dod (6)*, *Internet (1)*, *management (2)*, *mib-2(1)*, and *ip (4)*. This path is identified by (1.3.6.1.2.1.4) to indicate all the labeled numbers from the root to the *ip (4)* entry. Besides that entry, MIB module represents a number of network interfaces and well-known Internet protocols at the bottom of this tree. This path clearly shows all the standards of "IP" associated with the "MIB-2" computer networking "management."

9.7.4 Simple Network Management Protocol (SNMP)

The *Simple Network Management Protocol* (SNMP) is designed to monitor the performance of network protocols and devices. SNMP protocol data units (PDUs) can be carried in the payload of a UDP datagram, and so its delivery to a destination is not guaranteed. Managed devices, such as routers and hosts, are objects, and each object has a formal ASN.1 definition. For each object, MIB accommodates a database of information that describes its characteristics. With this protocol, a network manager can find the location of a fault. SNMP runs on top of UDP and uses client/server configurations. The commands of this protocol define how to query information from a server and forward information to a server or a client.

The task of SNMP is to transport MIB information among managing centers and agents executing on behalf of managing centers. For each managed MIB object, an SNMP request is used to retrieve or change its associated value. If an unsolicited message is received by an agent, or when an interface or device goes down, the protocol can also inform the managing center. The second version of this protocol, SNMPv2, runs on top of more protocols and has more messaging options, resulting in more effective network management. SNMPv3 has more security options.

SNMPv2 has seven PDUs, or messages, as follows.

1. GetRequest is used to obtain a MIB object value.
2. GetNextRequest is used to obtain the next value of a MIB object.
3. GetBulkRequest gets multiple values, equivalent to multiple GetRequests but without using multiple overheads.

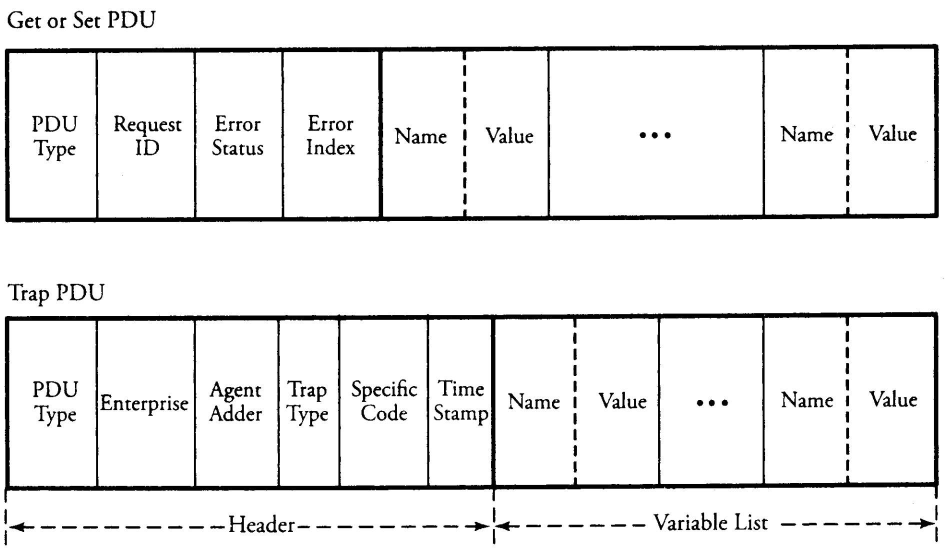


Figure 9.12 SNMP PDU format

4. InformRequest is a manager-to-manager message that two communicating management centers are remote to each other.
5. SetRequest is used by a managing center to initiate the value of a MIB object.
6. Response is a reply message to a request-type PDU.
7. Trap notifies a managing center that an unexpected event has occurred.

Figure 9.12 shows the format of SNMP PDUs. Two types of PDUs are depicted: Get or Set and Trap. The Get or Set PDU format is as follows:

- *PDU type* indicates one of the seven PDU types.
- *Request ID* is an ID used to verify the response of a request. Thus, a managing center can detect lost requests or replies.
- *Error status* is used only by Response PDUs to indicate types of errors reported by an agent.
- *Error index* is a parameter indicating to a network administrator which *name* has caused an error.

If requests or replies are lost, SNMP does not mandate any method for retransmission. *Error status* and *Error index* fields are all zeros except for the one in a GetBulkRequest PDU. Figure 9.12 also shows the format of the Trap PDU, whereby the *enterprise* field is for use in multiple networks; the *timestamp* field, for measuring up time; and the *agent address* field, for indicating that the address of the managed agent is included in the PDU header.

9.8 Summary

The top layer of the networking stack is known as the application layer. Services in this category allow users and programs to interact with services on remote machines and with remote users. The *application layer* offers network services and applications to users and runs certain applications. The *Domain Name System* (DNS) is a distributed hierarchical and global directory that translates machine or domain names to numerical IP addresses. An IP address can be assigned a *domain name*. Unique domain names assigned to hosts must be selected from a *name space* and are generally organized in a hierarchical fashion.

The TELNET remote login protocol uses a negotiation method to allow clients and servers to reconfigure the parameters controlling their interaction. The *Secure Shell* (SSH) remote login protocol is implemented on TCP for communications. SSH is more powerful and flexible than TELNET and allows the user to more easily execute a single command on a remote client.

The *Simple Mail Transfer Protocol* (SMTP) can transfer *e-mail* from the mail server of a source to mail servers of destinations. A user mailbox is a certain space the mail server allocates to the user to keep its e-mail. SMTP is designed to connect only the two mail servers of the associated parties, regardless of the distance between the two users. Consequently, this protocol involves only the two mail servers of the communicating users.

File transfer allows geographically distributed files and information to be shared among the members of a working group and for information sharing. A user can use a file transfer protocol to access a server and transfer the desired file. Two such protocols are FTP and SCP. The *World Wide Web* (WWW), or simply the *Web*, is a global network of servers linked together by a common protocol allowing acc ss to all connected hypertext resources. HTTP requests are first directed to the network *proxy server* called *Web cache*. Once configured by the network, a browser's request for an object is directed to the Web cache.

The chapter ended with network management aspects of computer networks. Managed devices, such as routers and hosts, are managed objects, and each object has a formal ASN.1 definition. Another tool through which a database of information and characteristics for objects can be accommodated is MIB. With SNMP, a network manager can find the location of a fault. SNMP runs on top of UDP and uses client/server configurations. SNMP commands define how to query information from a server and forward information to a server or a client.

The next chapter discusses the security aspects of computer networking. Types of network attacks, message encryption protocols, and message authentication techniques are covered.

9.9 Exercises

1. Find out about an IP address that is well known to you, such as that of your college or your work server. Set up an experiment in which you look up the domain name for this IP address.
2. Consider DNS servers.
   (a) Compare two approaches, obtaining a name from a file in a remote machine and from a DNS server of the local ISP.
   (b) Describe the relationship between a domain name taken from a DNS server and an IP address subnet.
   (c) In part (b), do all hosts on the same subnet need to be identified by the same DNS server? Why?
3. Read all the RFCs associated with TELNET and SSH listed in Appendix B.
   (a) What are the most significant differences between these two protocols?
   (b) Compare the two protocols on the functionality given by "rlogin."
4. Read all the RFCs associated with FTP listed in Appendix B.
   (a) Does FTP compute any checksum for files it transfers? Why?
   (b) FTP is based on TCP. Explain the status of file transfer if the TCP connection is shut down, and compare it with the same situation in which its control connection remains active.
   (c) Find all the commands from RFCs set for clients.
5. Set up an experiment in which two computers across a defined network exchange the same defined file through FTP. Measure the file transfer delay
   (a) On both directions when the network is in its best traffic state
   (b) On both directions when the network is in its worst traffic state
   (c) On one direction when you try FTP from one of the computers to itself

6. Do research on which letters, such as A, B, C, . . . , or signs, such as ♯, ∗, •, . . . , are not allowed to be sent on a URL to a Web server.
7. Read all the RFCs associated with HTTP listed in Appendix B.
   (a) What is the purpose of GET command?
   (b) What is the purpose of PUT command?
   (c) Following part (a), explain why a GET command needs to use the name of the contacted server when it is applied.
8. Figure 9.11 shows the hierarchical ASN.1 object identifier of network management.
   (a) Explain the role of ASN.1 in the protocol reference model, discussing it on either the five- or seven-layer protocol model.
   (b) Find the impact of constructing a grand set of unique global ASN.1 names for MIB variables.
   (c) Do research in the related RFCs listed in Appendix B, and find out where in the ASN.1 hierarchy a U.S. based organization must register its own developed MIB.
9. Consider the SNMP network management environment.
   (a) Justify why UDP is preferable as TCP for this environment.
   (b) MIB variables can be read in a local router the MIB belongs to. What would be the pros and cons if we let all managing centers access MIBs in all routers?
   (c) Should MIB variables be organized in the local router memory the same way that SNMP expresses? Why?
10. *Computer simulation project*. Write a computer program to simulate a TCP connection for a client/server combination. The server receives a small file from the client. Set up another machine as the proxy server, and try to send messages from the proxy server to the main server. Display all possible messages in the server.

CHAPTER 10

Network Security

The invention of computers made the need for security of digital information a critical issue. Advances in the field of computer networks have made information security even more important. Computer systems have to be equipped with mechanisms for securing data. Computer networks need provisions that secure data from possible intrusions. Security is especially crucial in wireless networks, as a wireless medium, by its nature, is vulnerable to intrusions and attacks. This chapter focuses on *network security*. The following major topics in network security are covered:

- *Overview of network security: elements and threats*
- *Overview of security methods*
- *Secret-key encryption protocols*
- *Public-key encryption protocols*
- *Authentication: message digest and digital signatures*
- *Security of IP and wireless networks*
- *Firewalls*

We begin by identifying and classifying types of network threats, hackers, and attacks, including DNS hacking attacks and router attacks. Network security can be divided into two broad categories: *cryptographic techniques* and *authentication techniques* (verification). Both secret-key and public-key encryption protocols are presented. We then discuss message authentication and digital signature methods, through which a receiver can be assured that an incoming message is from whom it says it is.

We then consider several standardized security techniques, such as IPsec, and the security of wireless networks and IEEE 802.11 standards, as well as firewalls, or devices designed to protect a network. Finally, we present a case study on the security of wireless networks.

10.1 Overview of Network Security

Network security is a top-priority issue in data networks. As communication networks are growing rapidly, security issues have pushed to the forefront of concern for end users, administrators, and equipment suppliers. Despite enormous joint efforts by various groups to develop effective security solutions for networks, hackers continue to pose new, serious threats by taking advantage of weaknesses present in the Internet infrastructure.

10.1.1 Elements of Network Security

Network security is concerned mainly with the following two elements:

1. *Confidentiality*. Information should be available only to those who have rightful access to it.
2. *Authenticity and integrity*. The sender of a message and the message itself should be verified at the receiving point.

In Figure 10.1, user 1 sends a message ("I am user 1") to user 2. In part (a) of the figure, the network lacks any security system, so an intruder can receive the message, change its content to a different message ("Hi! I am user 1") and send it to user 2. User 2 may not know that this falsified message is really from user 1 (authentication) and that the content of the message is what user 1 (confidentiality). In part (b) of the figure, a security block is added to each side of the communication, and a secret key that only users 1 and 2 would know about is included. Therefore, the message is changed to a form that cannot be altered by the intruder, who would be disabled in this communication transaction.

In general, no protocol or network architecture can ensure full security. Internet routing is based on a distributed system of many routers, switches, and protocols. These protocols have a number of points of vulnerabilities that can be exploited to cause such

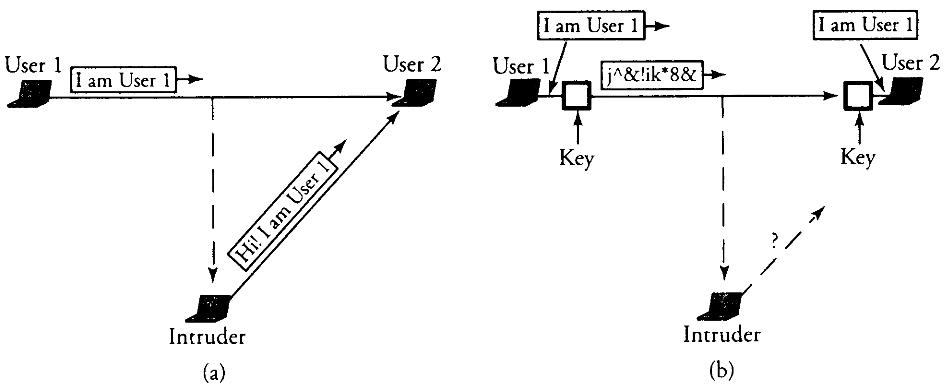


Figure 10.1 (a) Message content and sender identity falsified by intruder; (b) a method of applied security

problems as misdelivery or nondelivery of user traffic, misuse of network resources, network congestion and packet delays, and the violation of local routing policies.

10.1.2 Threats to Network Security

Internet infrastructure *attacks* are broadly classified into four categories, as follows:

1. DNS hacking
2. Routing table poisoning
3. Packet mistreatment
4. Denial of service

Among these threats, the first three attacks are related to network infrastructure; ***denial-of-service attacks*** are related to end systems.

DNS Hacking Attacks

As mentioned in Chapter 9, the ***Domain Name System*** (DNS) server is a distributed hierarchical and global directory that translates domain names into numerical IP address. DNS is a critical infrastructure, and all hosts contact DNS to access servers and start connections. In the normal mode of operation, hosts send UDP queries to the DNS server. Servers reply with a proper answer, or direct the queries to smarter servers. A DNS server also stores information other than host addresses.

Name-resolution services in the modern Internet environment are essential for e-mail transmission, navigation to Web sites, or data transfer. Thus, an attack on DNS can potentially affect a large portion of the Internet. A DNS *hacking attack* may result in the lack of data authenticity and integrity and can appear in any of the following forms:

1. An *information-level attack* forces a server to correspond with other than the correct answer. With cache poisoning, a hacker tricks a remote name server into caching the answer for a third-party domain by providing malicious information for the domain's authorized servers. Hackers can then redirect traffic to a preselected site.
2. In a *masquerading attack*, the adversary poses as a trusted entity and obtains all the secret information. In this guise, the attacker can stop any message from being transmitted further or can change the content or redirect the packet to bogus servers. This action is also known as a *middle-man attack*.
3. The attacker normally sends queries to each host and receives in reply the DNS host name. In an *information leakage attack*, the attacker sends queries to all hosts and identifies which IP addresses are not used. Later on, the intruder can use those IP addresses to make other types of attacks.
4. Once a domain name is selected, it has to be registered. Various tools are available to register domain names over the Internet. If the tools are not smart enough, an invader might obtain secure information and use it to highjack the domain later. In the *domain highjacking attack*, whenever a user enters a domain address, she/he is forced to enter into the attacker's Web site. This can be very irritating and can cause a great loss of Internet usage ability.

Routing Table Poisoning Attacks

A *routing table poisoning attack* is the undesired modification of routing tables. An attacker can do this by maliciously modifying the routing information update packets sent by routers. This is a challenging and important problem, as a routing table is the basis of routing in the Internet. Any false entry in a routing table could lead to significant consequences, such as congestion, an overwhelmed host, looping, illegal access to data, and network partition. Two types of routing table poisoning attacks are the *link attack* and the *router attack*.

A *link attack* occurs when a hacker gets access to a link and thereby intercepts, interrupts, or modifies routing messages on packets. Link attacks act similarly on both the link-state and the distance-vector protocols discussed in Chapter 7. If an attacker succeeds in placing an attack in a link-state routing protocol, a router may send incorrect updates about its neighbors or remain silent even if the link state of its neighbor has changed. The attack through a link can be so severe that the attacker can program a router to either drop packets from a victim or readdress packets to a victim, resulting in a lower throughput of the network. Sometimes, a router can stop an intended packet from being forwarded further. However, since more than one path to any destination exists, the packet ultimately reaches its destination.

Router attacks may affect the link-state protocol or even the distance-vector protocol. If link-state protocol routers are attacked, they become malicious. They may add a nonexisting link to a routing table, delete an existing link, or even change the cost of a link. This attack may cause a router to simply ignore the updates sent by its neighbors, leading to a serious impact on the operability of the network traffic flow.

In the distance-vector protocol, an attacker may cause routers to send wrong updates about any node in the network, thereby misleading a router and resulting in network problems.

Most unprotected routers have no way to validate updates. Therefore, both link-state and distance-vector router attacks are very effective. In the distance-vector protocol, for example, a malicious router can send wrong information in the form of a distance vector to all its neighbors. A neighbor may not be able to detect this kind of attack and thus proceeds to update its routing table, based on wrong distance vectors. The error can in turn be propagated to a great portion of the network before being detected.

Packet-Mistreatment Attacks

A *packet-mistreatment attack* can occur during any data transmission. A hacker may capture certain data packets and mistreat them. This type of attack is very difficult to detect. The attack may result in congestion, lowering throughput, and denial-of-service attacks. Similar to routing table poisoning attacks, packet-mistreatment attacks can also be subclassified into *link attacks* and *router attacks*. The link attack causes interruption, modification, or replication of data packets. A router attack can misroute all packets and may result in congestion or denial of service. Following are some examples of a packet-mistreatment attack:

- *Interruption.* If an attacker intercepts packets, they may not be allowed to be propagated to their destinations, resulting in a lower throughput of the network. This kind of attack cannot be detected easily, as even in normal operations, routers can drop some packets, for various reasons.
- *Modification.* Attackers may succeed in accessing the content of a packet while in transit and change its content. They can then change the address of the packet or

even change its data. To solve this kind of problem, a digital signature mechanism, discussed later in this chapter, can be used.

- *Replication*. An attacker might trap a packet and replay it. This kind of attack can be detected by using the sequence number for each packet.
- *Ping of death*. An attacker may send a *ping message*, which is large and therefore must be fragmented for transport. The receiver then starts to reassemble the fragments as the ping fragments arrive. The total packet length becomes too large and might cause a system crash.
- *Malicious misrouting ofpackets*. A hacker may attack a router and change its routing table, resulting in misrouting of data packets, causing a denial of service.

Denial-of-Service Attacks

A *denial-of-service attack* is a type of security breach that prohibits a user from accessing normally provided services. The denial of service does not result in information theft or any kind of information loss but can nonetheless be very dangerous, as it can cost the target person a large amount of time and money. Denial-of-service attacks affect the destination rather than a data packet or router.

Usually, a denial-of-service attack affects a specific network service, such as e-mail or DNS. For example, such an attack may overwhelm the DNS server in various ways and make it inoperable. One way of initiating this attack is by causing buffer overflow. Inserting an executable code inside memory can potentially cause a buffer overflow. Or, an adversary may use various tools to send large numbers of queries to a DNS server, which then is not able to provide services in a timely manner.

Denial-of-service attacks are easy to generate but difficult to detect. They take important servers out of action for few hours, thereby denying service to all users. There are yet a few other situations that can cause this kind of attack, such as UDP flood, a TCP flood and ICMP flood. In all these attacks, the hacker's main aim is to overwhelm victims and disrupt services provided to them.

Denial-of-service attacks are two types:

1. *Single-source*. An attacker sends a large number of packets to a target system to overwhelm and disable it. These packets are designed such that their real sources cannot be identified.
2. *Distributed*. In this type of attack, a large number of hosts are used to flood unwanted traffic to a single target. The target cannot then be accessible to other users in the network, as it is processing the flood of traffic.

The flood may be either a UDP flood or a TCP SYN flood. UDP flooding is used against two target systems and can stop the services offered by either system. Hackers link the UDP character-generating services of a system to another one by sending UDP packets with spoofed return addresses. This may create an infinite looping between the two systems, leading to system uselessness.

Normally, a SYN packet is sent by a host to a user who intends to establish a connection. The user then sends back an acknowledgment. In the TCP SYN flood, a hacker sends a large number of SYN packets to a target user. Since the return addresses are spoofed, the target user queues up a SYN/ACK packet and never processes it. Therefore, the target system keeps on waiting. The result may be a hard disk crash or reboot.

10.2 Overview of Security Methods

Common solutions that can protect computer communication networks from attacks are classified as *cryptographic techniques* or *authentication techniques* (verification).

10.2.1 Cryptographic Techniques

Cryptography has a long and fascinating history. Centuries ago, cryptography was used as a tool to protect national secrets and strategies. Today, network engineers focus on *cryptography* methods for computer communication networks. Cryptography is the process of tranforming a piece of information or message shared by two parties into some sort of code. The message is scrambled before transmission so that it is undetectable by outside watchers. This kind of message needs to be decoded at the receiving end before any further processing.

The main tool that network security experts are using to encrypt a message M is a secret key K; the fundamental operation often used to encrypt a message is the Exclusive-OR ($\oplus$). Suppose that we have one bit, M, and a secret bit, K. A simple encryption is carried out using $M \oplus K$. To decrypt this message, the second party—if he/she has the key, K—can easily detect M by performing the following:

$$(M \oplus K) \oplus K = M. \tag{10.1}$$

In computer communication networks, data can travel between two users while it is encrypted. In Figure 10.2, two servers are exchanging data while two types of encryption devices are installed in their communication network. The first encryption device is *end-to-end encryption*, whereby secret coding is carried out at both end systems.

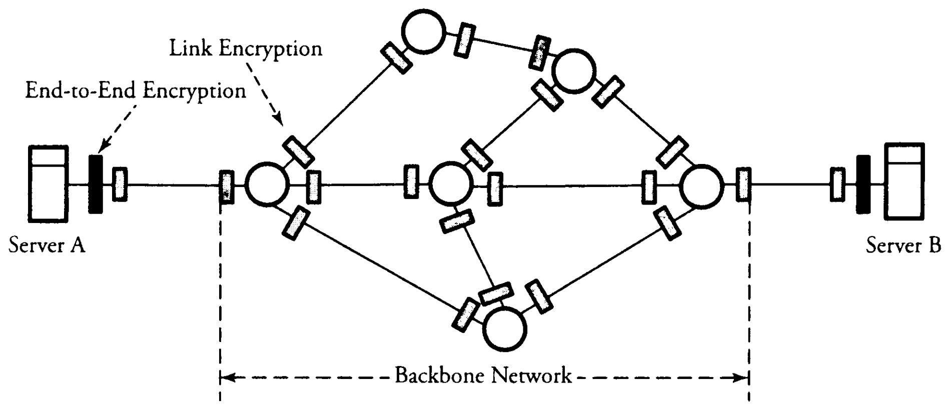


Figure 10.2 Overview of encryption points in a communication network

In this figure, server *A* encodes its data, which can be decoded only at the other end server. The second phase of encryption is *link encryption*, which secures all the traffic passing over that link.

The two types of encryption techniques are *secret-key encryption* and *public-key encryption*. In a secret-key model, both sender and receiver conventionally use the same key for an encryption process. In a public-key model, a sender and a receiver each use a different key. The public-key system is more powerful than the secret-key system and provides better security and message privacy . But the biggest drawback of public-key encryption is speed. The public-key system is significantly more complex computationally and may not be practical in many cases. Hence, the public-key system is used only to establish a session to exchange a session key. Then, this session key is used in a secret-key system for encrypting messages for the duration of the session.

10.2.2 Authentication Techniques

Encryption methods offer the assurance of message confidentiality. However, a networking system must be able to verify the authenticity of the message and the sender of the message. These forms of security techniques in computer networks are known as *authentication techniques* and are categorized as *authentication with message digest* and *authentication with digital signature.* Message authentication protects a user in a network against data falsification and ensures data integrity. These methods do not necessarily use keys.

10.3 Secret-Key Encryption Protocols

Secret-key encryption protocols, sometimes known as *symmetric encryption*, or *single-key encryption* protocols, are conventional encryption models. They typically consist of an encryption algorithm, a key, and a decryption algorithm. At the end point, the encrypted message is called *ciphertext*. Several standard mechanisms can be used to implement a secret-key encryption algorithm. Here, we focus on two protocols: *Data Encryption Standard* (DES) and *Advanced Encryption Standard* (AES).

In these algorithms, a shared secret key between a transmitter and a receiver is assigned at the transmitter and receiver points. The encryption algorithm produces a different key at any time for a specific transmission. Changing the key changes the output of the algorithm. At the receiving end, the encrypted information can be transformed back to the original data by using a decryption algorithm and the same key that was used for encryption. The security of conventional encryption depends on the secrecy of the key, not on the secrecy of the encryption algorithm. Consequently, the algorithm need not be kept secret; only the key has to be secret.

10.3.1 Data Encryption Standard (DES)

With the *Data Encryption Standard* (DES), plaintext messages are converted into 64-bit blocks, each encrypted using a key. The key length is 64 bits but contains only 56 usable bits; thus, the last bit of each 8 byte in the key is a parity bit for the corresponding byte. DES consists of 16 identical rounds of an operation, as shown in Figure 10.3. The details of the algorithm on each 64-bit block of message at each round i of operation are as follows.

Begin DES Algorithm

1. **Initialize.** Before round 1 begins, all 64 bits of an incoming message and all 56 bits of the secret key are separately permuted (shuffled).
2. Each incoming 64-bit message is broken into two 32-bit halves denoted by L_i and R_i, respectively .
3. The 56 bits of the key are also broken into two 28-bit halves, and each half is rotated one or two bit positions, depending on the round.
4. All 56 bits of the key are permuted, producing version k_i of the key on round i.

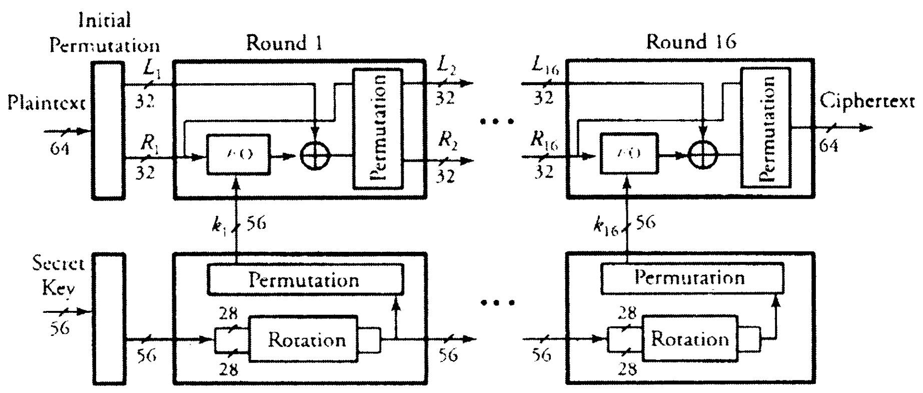


Figure 10.3 The Data Encryption Standard (DES)

5. In this step, $\oplus$ is a logic Exclusive-OR, and the description of function $F()$ appears next. Then L_i and R_i are determined by

$$L_i = R_{i-1} \tag{10.2}$$

and

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i). \tag{10.3}$$

6. All 64 bits of a message are permuted. ■

The operation of function $F()$ at any around i of DES is as follows.

1. Out of 56 bits of k_i, function $F()$ chooses 48 bits.
2. The 32-bit R_{i-1} is expanded from 32 bits to 48 bits so that it can be combined with 48-bit k_i. The expension of R_{i-1} is carried out by first breaking R_{i-1} into eight 4-bit chunks and then expanding each chunk by copying the leftmost bit and the rightmost bit from right and left adjacent chunks, respectively.
3. Function $F()$ also partitions the 48 bits of k_i into eight 6-bit chunks.
4. The corresponding eight chunks of R_{i-1} and eight chunks of k_i are combined as follows:

$$R_{i-1} = R_{i-1} \oplus k_i. \tag{10.4}$$

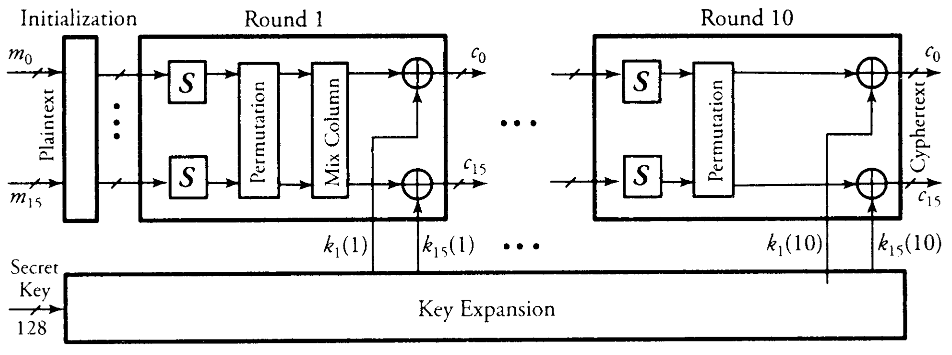


Figure 10.4 Overview of Advanced Encryption Standard (AES) protocol

At the receiver, the same steps and the same key are used to reverse the encryption. It is now apparent that the 56-bit key length may not be sufficient to provide full security. This argument is still controversial. Triple DES provides a solution for this controversy: three keys are used, for a total of 168 bits. It should also be mentioned that DES can be implemented more efficiently in hardware than in software.

10.3.2 Advanced Encryption Standard (AES)

The *Advanced Encryption Standard* (AES) protocol has a better security strength than DES. AES supports 128-bit symmetric block messages and uses 128-, 192-, or 256-bit keys. The number of rounds in AES is variable from 10 to 14 rounds, depending on the key and block sizes. Figure 10.4 illustrates the encryption overview of this protocol, using a 128-bit key. There are ten rounds of encryptions for the key size of 128 bits. All rounds are identical except for the last round, which has no mix-column stage.

A single block of 128-bit plaintext (16 bytes) as an input arrives from the left. The plaintext is formed as 16 bytes m_0 through m_{15} and is fed into round 1 after an initialization stage. In this round, substitute units—indicated by S in the figure—perform a byte-by-byte substitution of blocks. The ciphers, in the form of rows and columns, move through a *permutation stage* to shift rows to mix columns. At the end of this round, all 16 blocks of ciphers are Exclusive-ORed with the 16 bytes of round 1 key $k_0(1)$ through $k_{15}(1)$. The 128-bit key is expanded for ten rounds. The AES *decryption algorithm* is fairly simple and is basically the reverse of the encryption algorithm at each stage of a round. All stages of each round are reversible.

10.4 Public-Key Encryption Protocols

The introduction of *public-key encryption* brought a revolution to the field of cryptography. Public-key cryptography provided a very clever method for key exchange. In the public-key encryption model, a sender/receiver pair use different keys. This model is sometimes known as *asymmetric*, or *two-key*, *encryption*.

Public-key algorithm is based on mathematical functions rather than on substitution or permutation, although the security of any encryption scheme indeed depends on the length of the key and the computational work involved in breaking an encrypted message. Several public-key encryption protocols can be implemented. Among them, the following two protocols are the focus of our study:

- *Rivert, Shamir, and Aldeman* (RSA) protocol
- *Diffie-Hillman key-exchange* protocol.

In the public-key encryption methods, either of the two related keys can be used for encryption; the other one, for decryption. It is computationally infeasible to determine the decryption key given only the algorithm and the encryption key. Each system using this encryption method generates a pair of keys to be used for encryption and decryption of a message that it will receive. Each system publishes its encryption key by placing it in a public register or file and sorts out the key as a public one.

The companion key is kept private. If A wishes to send a message to B, A encrypts the message by using B's public key. On receiving the message, B decrypts it with the private B key. No other recipients can decrypt the message, since only B knows its private key. This way, public-key encryption secures an incoming communication as long as a system controls its private key. However, public-key encryption has extra computational overhead and is more complex than the conventional one.

10.4.1 RSA Algorithm

Rivert, Shamir, and Aldeman developed the RSA public-key encryption and signature scheme. This was the first practical public-key encryption algorithm. RSA is based on the intractability of factoring large integers. Assume that a plaintext m must be encrypted to a ciphertext c. The RSA algorithm has three phases for this: *key generation*, *encryption*, and *decryption*.

Key Generation

In the RSA scheme, the key length is typically 512 bits, which requires an enormous computational power. A plaintext is encrypted in blocks, with each block having a binary value less than some number n. Encryption and decryption are done as follows, beginning with the generation of a public key and a private key.

Begin Key Generation Algorithm

1. Choose two roughly 256-bit prime numbers, a and b, and derive $n = ab$. (A number is prime if it has factors of 1 and itself.)
2. **Find** x. Select encryption key x such that x and $(a-1)(b-1)$ are relatively prime. (Two numbers are relatively prime if they have no common factor greater than 1.)
3. **Find** y. Calculate decryption key y:

$$xy \bmod (a-1)(b-1) = 1. \tag{10.5}$$

4. At this point, a and b can be discarded.
5. The public key = $\{x, n\}$.
6. The private key = $\{y, n\}$. ■

In this algorithm, x and n are known to both sender and receiver, but only the receiver must know y. Also, a and b must be large and about the same size and both greater than 1,024 bits. The larger these two values, the more secure the encryption.

Encryption

Both sender and receiver must know the value of n. The sender knows the value of x, and only the receiver knows the value of y. Thus, this is a public-key encryption, with the public key $\{x, n\}$ and the private key $\{y, n\}$. Given $m < n$, ciphertext c is constructed by

$$c = m^x \bmod n. \tag{10.6}$$

Note here that if a and b are chosen to be on the order of 1,024 bits, $n \approx 2,048$. Thus, we are not able to encrypt a message longer than 256 characters.

Decryption

Given the ciphertext, c, the plaintext, m, is extracted by

$$m = c^y \bmod n. \tag{10.7}$$

In reality, the calculations require a math library, as numbers are typically huge. One can see easily how Equations (10.6) and (10.7) work.

Example. For an RSA encryption of a 4-bit message of 1,001, or $m = 9$, we choose $a = 3$ and $b = 11$. Find the public and the private keys for this security action, and show the ciphertext.

Solution. Clearly, $n = ab = 33$. We select $x = 3$, which is relatively prime to $(a - 1)(b - 1) = 20$. Then, from $xy \bmod (a - 1)(b - 1) = 3y \bmod 20 = 1$, we can get $y = 7$. Consequently, the public key and the private key should be {3, 33} and {7, 33}, respectively. If we encrypt the message, we get $c = m^x \bmod n = 9^3 \bmod 33 = 3$. The decryption process is the reverse of this action, as $m = c^y \bmod n = 3^7 \bmod 33 = 9$.

10.4.2 Diffie-Hillman Key-Exchange Protocol

In the *Diffie-Hillman key-exchange* protocol, two end users can agree on a shared secret code without any information shared in advance. Thus, intruders would not be able to access the transmitted communication between the two users or discover the shared secret code. This protocol is normally used for *virtual private networks* (VPNs), explained in Chapter 16. The essence of this protocol for two users, 1 and 2, is as follows. Suppose that user 1 selects a prime a, a random integer number x_1, and a generator g and creates $y_1 \in \{1, 2, \cdots, a - 1\}$ such that

$$y_1 = g^{x_1} \bmod a. \tag{10.8}$$

In practice, the two end users agree on a and g ahead of time. User 2 performs the same function and creates y_2:

$$y_2 = g^{x_2} \bmod a. \tag{10.9}$$

User 1 then sends y_1 to user 2. Now, user 1 forms its key, k_1, using the information its partner sent as

$$k_1 = y_2^{x_1} \bmod a, \tag{10.10}$$

and user 2 forms its key, k_2, using the information its partner send it as

$$k_2 = y_1^{x_2} \bmod a. \tag{10.11}$$

It can easily be proved that the two Keys k_1 and k_2 are equal. Therefore, the two users can now encrypt their messages, each using its own key created by the other one's information.

10.5 Authentication

Authentication techniques are used to verify identity. Message authentication verifies the authenticity of both the message content and the message sender. Message content is authenticated through implementation of a hash function and encryption of the resulting message digest. The sender's authenticity can be implemented by use of a digital signature.

A common technique for authenticating a message is to implement a *hash function*, which is used to produce a "fingerprint" of a message. The hash value is added at the end of message before transmission. The receiver recomputes the hash value from the received message and compares it to the received hash value. If the two hash values are the same, the message was not altered during transmission. Once a hash function is applied on a message, m, the result is known as a message digest, or $h(m)$. The hash function has the following properties.

- Unlike the encryption algorithm, the authentication algorithm is not required to be reversible.
- Given a message digest $h(m)$, it is computationally infeasible to find m.
- It is computationally infeasible to find two different messages m_1 and m_2 such that $h(m_1) = h(m_2)$.

Message authentication can be implemented by two methods. In the first method, as shown in Figure 10.5 (a), a hash function is applied on a message, and then a process of encryption is implemented. Thus, a message digest can also be encrypted in this method. At this stage, the encryption can be a public key or a secret key. The authenticity of a message in this method is assured only if the sender and the receiver share the encryption key. At the receiver site, the receiving user 2 has to decrypt the received message digest and compare it with the one made locally at its site for any judgments on the integrity of the message.

In the second method, as shown in Figure 10.5 (b), no encryption is involved in the process of message authentication. This method assumes that the two parties share a secret key. Hence, at the receiving site, the comparison is made between the received $h(m)$ and the message digest made locally from the received message. This technique is more popular in the security infrastructure of the Internet Protocol. Among the message authentication protocols are the MD5 *hash algorithm* and the *Secure Hash Algorithm* (SHA). SHA is the focus of our discussion.

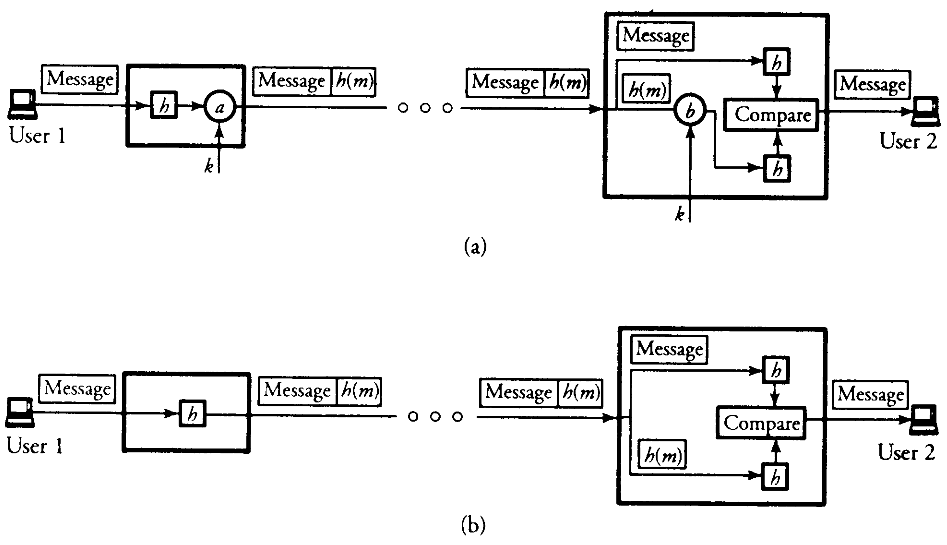


Figure 10.5 Message authentication: (a) combined with encryption; (b) use of the hash function

10.5.1 Secure Hash Algorithm (SHA)

The *Secure Hash Algorithm* (SHA) was proposed as part of the digital signature standard. SHA-1, the first version of this standard, takes messages with a maximum length of 2^{24} and produces a 160-bit digest. With this algorithm, SHA-1 uses five registers, R_1 through R_5, to maintain a "state" of 20 bytes.

The first step is to pad a message m with length ℓ_m. The message length is forced to $\ell_m = 448 \bmod 512$. In other words, the length of the padded message becomes 64 bits less than the multiple of 512 bits. The number of padding bits can be as low as 1 bit and as high as 512 bits. The padding includes a 1 bit and as many 0 bits as required. Therefore, the least-significant 64 bits of the message length are appended to convert the padded message to a word with a multiple of 512 bits.

After padding, the second step is to expand each block of 512-bit (16 32 bits) words $\{m_0, m_1, \cdots, m_{15}\}$ to words of 80 32 bits using:

$$w_i = m_i \text{ for } 0 \le i \le 15 \tag{10.12}$$

and

$$w_i = w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} \hookleftarrow 1 \text{ for } 16 \le i \le 79, \tag{10.13}$$

where $\hookleftarrow j$ means left rotation by j bits. This way, bits are shifted several times if the incoming block is mixed with the state. Next, bits from each block of w_i are mixed into the state in four steps, each maintaining 20 rounds. For any values of a, b, and c, and bit number i, we define a function $F_i(a, b, c)$ as follows:

$$F_i(a, b, c) = \begin{cases} (a \cap b) \cup (\bar{a} \cap c) & 0 \le i \le 19 \\ a \oplus b \oplus c & 20 \le i \le 39 \\ (a \cap b) \cup (a \cap c) \cup (b \cap c) & 40 \le i \le 59 \\ a \oplus b \oplus c & 60 \le i \le 79 \end{cases} \tag{10.14}$$

Then, the 80 steps ($i = 0, 1, 2, \cdots, 79$) of the four rounds are described as follows:

$$\delta = (R_1 \hookleftarrow 5) + F_i(R_2, R_3, R_4) + R_5 + w_i + C_i \tag{10.15}$$

$$R_5 = R_4 \tag{10.16}$$

$$R_4 = R_3 \tag{10.17}$$

$$R_3 = R_2 \hookleftarrow 30 \tag{10.18}$$

$$R_2 = R_1 \tag{10.19}$$

$$R_1 = \delta, \tag{10.20}$$

where C_i is a constant value specified by the standard for round i. The message digest is produced by concatenation of the values in R_1 through R_5.

10.6 Authentication and Digital Signature

A *digital signature* is one of the most important required security measures. Much like a person's signature on a document, a digital signature on a message is required for the authentication and identification of the right sender. The digital signature is supposed to be unique to an individual and serves as a means of identifying the sender. An electronic signature is not as easy as it was with the paper-based system. The digital signature requires a great deal of study, research, and skill. Even if these requirements are met, there is no guarantee that all the security requirements have been met.

The technical method of providing a sender's authentication is performed through cryptography. Many cryptographic mechanisms have been developed. Among them, the RSA algorithm implements both encryption and digital signature. When RSA is applied, the message is encrypted with the sender's private key. Thus, the entire encrypted message serves as a digital signature. This means that at the receiving end, the receiver can decrypt it, using the public key. This authenticates that the packet comes from the right user.

10.7 Security of IP and Wireless Networks

This section presents a case study of some of the security policies adopted for the Internet Protocol (IP) and basic wireless technologies. We start with IPsec, a standard for the IP layer.

10.7.1 IP Security and IPsec

Between any two users with a TCP/IP connection are multiple secure layers of security. A number of fields are appended to an IP packet when it is ready to undergo the security implementation. IP *security* (IPsec) is a set of protocols developed by the *Internet Engineering Task Force* (IETF) to support the secure exchange of packets at the IP layer. Figure 10.6 illustrates an encrypted and authenticated IP packet. An IPsec authentication header has the following fields.

- *Security parameters index* expresses a one-way relationship between two communicating users that can accept security.
- *Sequence number* is an increasing counter number.
- *Payload data* is an encryption-protected upper-layer segment.
- *Padding* is used if the plaintext is required by the encryption algorithm to be a certain multiple of 1 byte.
- *Pad length* specifies the number of bytes for the padding.
- *Next header* indicates the type of next header attached.
- *Authentication data* provides the integrity check value.

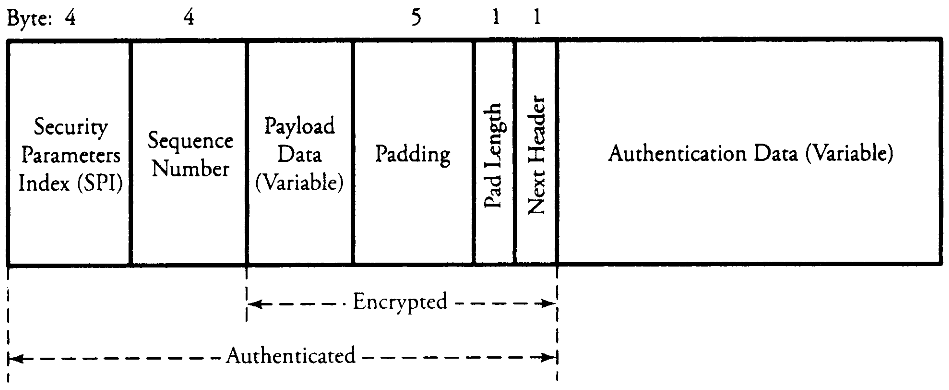


Figure 10.6 IPsec authentication header format

IPsec provides enhanced security features, such as better encryption algorithms and more comprehensive authentication. In order for IPsec to work, both sender and receiver must exchange public encryption keys. IPsec has two encryption modes: tunnel and transport. Tunnel mode encrypts the header and the payload of each packet; the transport mode encrypts the payload. IPsec can encrypt data between devices: router to router, security device to router, PC to router, and PC to server.

10.7.2 Security of Wireless Networks and IEEE 802.11

Wireless networks are particularly vulnerable because of their nonwired infrastructure. Intruders can access wireless networks by receiving radio waves carrying packets and frames propagated beyond the needed range of the network's base station and hosts. Our focus here is the security mechanisms for the wireless 802.11 standards known as *wired equivalent privacy* (WEP).

This section also describes types of security features desired for IEEE 802.11a, b, and i. WEP provides a level of security similar to that found in wired networks. It is a standard of security for IEEE 802.11a and b and offers authentication and data encryption between a host and a wireless base station, using a secret shared key. The essence of this protocol between a host and a base station (wireless access point) is as follows.

1. The host requests authentication from the base station.
2. The base station responds.
3. The host encrypts data by using secret-key encryption.
4. The base station decrypts the received encrypted data. If the decrypted data matches the original one sent to the host, the host is authenticated by the base station.

Figure 10.7 shows how data is encrypted. First, a 40-bit secret key, k, known by both the host and the base station, is created. A 24-bit initialization field to be used to encrypt a single frame is appended to this key. The initialization field is different for each frame.

As shown in the figure, a 4-byte CRC field is computed for the data payload. The payload and the CRC bytes are then encrypted. The encryption algorithm produces a stream of key values: $k_1, k_2, \cdots, k_i, \cdots, k_{n-1}, k_n$. Assume that the plaintext is partitioned into i bytes. Let c_i be the ith byte of the ciphertext and m_i be the ith byte of the plaintext; the encryption is done by using k_i, as follows:

$$c_i = m_i \oplus k_i. \tag{10.21}$$

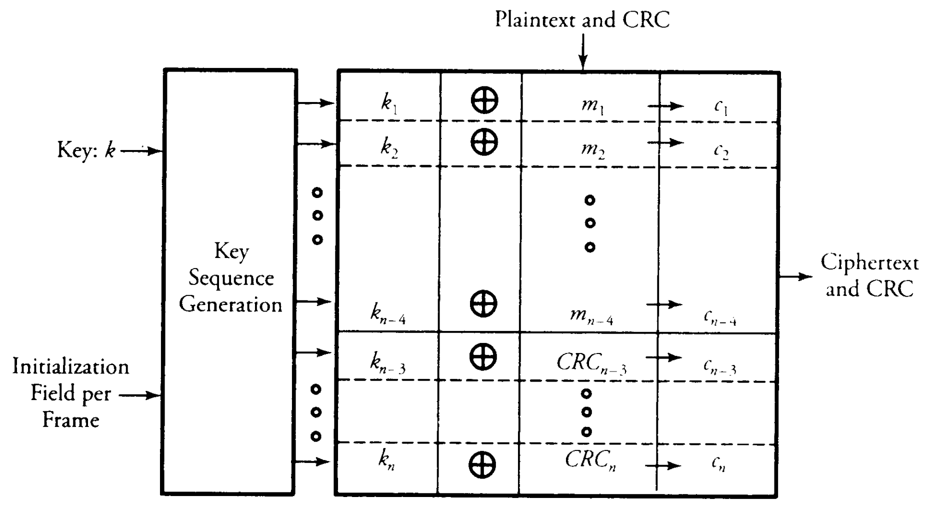


Figure 10.7 Security implementation in wireless IEEE 802.11

To decrypt the ciphertext, the receiver uses the same secret key as the sender used, appends the initialization field to it, and calculates

$$m_i = c_i \oplus k_i. \tag{10.22}$$

WEP is simple and relatively weak. The procedure for security of IEEE 802.11i is different because of its more sophisticated characteristics. This standard specifies an authentication server for the base-station communication. The separation of the authentication server from the base station means that the authentication server can serve many base stations. A new protocol, called *Extensible Authentication Protocol* (EAP), specifies the interaction between a user and an authentication server (IEEE 802.11i).

To summarize the IEEE 802.11i security mechanism: A base station first announces its presence and the types of security services it can provide to the wireless users. This way, users can request the appropriate type and level of encryption or authentication. EAP frames are encapsulated and sent over the wireless link. After decapsulation at the base station, the frames are encapsulated again, this time using a protocol called RADIUS for transmission over UDP to the authentication server. With EAP, public-key encryption is used, whereby a shared secret key known only to the user and the authentication server is created. The wireless user and the base station can also generate additional keys to perform the link-level encryption of data sent over the wireless link, which makes it much more secure than the one explained for 802.11a,b.

10.8 Firewalls

As the name suggests, a *firewall* protects data from the outside world. A firewall can be a software program or a hardware device. A firewall a popular security mechanism for networks. A firewall is a simple router implemented with a special program. This unit is placed between hosts of a certain network and the outside world, as shown in Figure 10.8, and the rest of the network. The security issues faced by a smaller network like the one used at home are similar to larger networks. A firewall is used to protect the network from unwanted Web sites and potential hackers.

A firewall is placed on the link between a network router and the Internet or between a user and a router. The objective of such a configuration is to monitor and filter packets coming from unknown sources. Consequently, hackers do not have access to penetrate through a system if a firewall protects the system. For a large company with many small networks, the firewall is placed on every connection attached to the Internet. Companies can set rules about how their networks or particular systems need to work in order to maintain security. Companies can also set rules on how a system can connect to Web sites. These precautionary rules are followed in order to attain the advantage of having a firewall. Hence, the firewall can control how a network works with an Internet connection. A firewall can also be used to control data traffic.

Software firewall programs can be installed in home computers by using an Internet connection with these so-called gateways, the computer with such a software can access Web servers only through this software firewall. But hardware firewalls are more secure than software firewalls. Moreover, hardware firewalls are not expensive. Some firewalls also offer virus protection. The biggest security advantage of installing a firewall in a business network is to protect from any outsider logging on to the network under protection. Firewalls are preferred for use in almost all network security infrastructures,

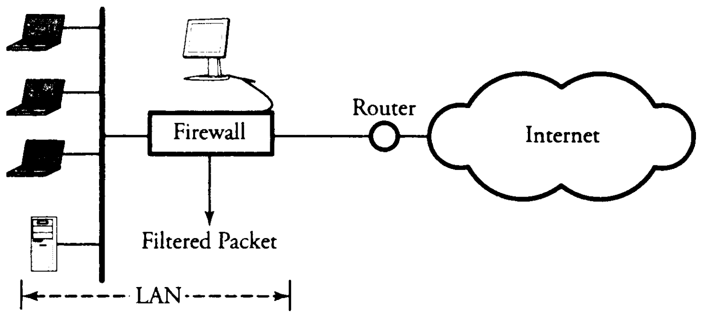


Figure 10.8 A simple configuration of a secured network using a firewall

as they allow the implementation of a security policy in one centralized place rather than end to end. Sometimes, a firewall is put where a large amount of data flow is normally expected.

A firewall controls the flow of traffic by one of the following three methods. The first method is *packet filtering*. Apart from forwarding packets between networks, a firewall filters those packets that pass through. If packets can get through the filter, they reach their destinations; otherwise, they are discarded. A firewall can be programmed to throw away certain packets addressed to a particular IP host or TCP port number. This condition is especially useful if a particular host does not want to respond to any access from an external source.

The second method is that a firewall filters packets based on the source IP address. This filtering is helpful when a host has to be protected from any unwanted external packets. The third method, denial of service, was explained earlier. This method controls the number of packets entering a network.

10.9 Summary

Network security is one of the top-priority issues in computer and communication networks. Security issues have pushed to the forefront of concern for end users, administrators, and equipment suppliers. Networking attacks can be classified as DNS *hacking*, *routing table poisoning*, *packet mistreatment*, and *denial of service*. Two major solutions for computer networking security are *cryptographic techniques* and *authentication techniques* (verification). The main tool that network security experts use to encrypt a message is a *key*, and the fundamental operation often used to encrypt a message is the Exclusive-OR operation.

Two secret-key encryption protocols are the *Data Encryption Standard* (DES), and the *Advanced Encryption Standard* (AES). In both methods, a secret key is shared between a transmitter and its receiver by assigning it to both the transmitter point and the receiver point. Public-key cryptography is more effective; a sender/receiver pair use different keys for encryption and decryption, and each party can publish its public (encryption) key so that anyone can use it to encrypt a message for that party. Two public-key protocols are the *Rivert, Shamir, and Aldeman* (RSA) protocol and the *Diffie-Hillman key-exchange* protocol.

A receiver can use *message authentication* methods to be assured that the incoming message is from its purported sender. Cryptographic hash functions are used in message authentication codes. Such codes can be generalized to produce a digital signature that guarantees a document's *authenticity*. The *Secure Hash Algorithm* (SHA) has been proposed as part of the digital signature standard.

The IP security (IPsec) protocol requires both sender and receiver to exchange public encryption keys. IPsec has two encryption modes: tunnel and transport. The tunnel mode encrypts the header and the payload of each packet, and the transport mode encrypts the payload. IPsec can encrypt data between router and router, security device and router, PC and router, and PC and server. Another security mechanism is a firewall, which is placed on the link between a network router and the Internet or between a user and a router. The objective of such a configuration is to filter packets coming from unknown sources.

Part II of the book follows. Chapter 11 presents analytical methods for delay estimations of single queues of packets and networks of single queues.

10.10 Exercises

1. Assume that in round 4 of a DES process for a message, L_4 = 4de5635d (in hex), R_4 = 3412a90e (in hex) and k_5 = be1142 7e6ac2 (in hex). Find R_5 and L_5.
2. Use DES to encrypt a 64-bit message, including all 1s with a 56-bit secret key consisting of 0101...01. Assuming a 1-bit rotation in the key process, find the outputs of the first round.
3. Check your system to find out about its encryption utility. If a DES algorithm is used:
   (a) Use an experiment to measure the speed of encryption and decryption.
   (b) Repeat part (a), using different keys.
4. Write a computer program to find the ciphertext for a message, using DES to encrypt a 64-bit message, including all 0s with a 56-bit secret key consisting of 1010...10. Assume that a 1-bit rotation in the key process.
5. In the RSA encryption algorithm, show how either of Equations (10.6) and (10.7) is concluded from the other one.
6. For an RSA encryption of a 4-bit message 1010, we chose $a = 5$, $b = 11$, and $x = 3$. Find the public and the private keys for this security action, and show the ciphertext.
7. Apply RSA and do the following.
   (a) Encrypt $a = 5$, $b = 11$, $x = 7$, and $m = 13$.

(b) Find the corresponding y.
(c) Decrypt the ciphertext

8. Normally the speed of the RSA encryption algorithm is much lower than secret-key encryption algorithms. To solve this issue, we can combine RSA with a secret-key encryption algorithm, such as AES. Assume that user 1 chooses an AES key of 256 bits and encrypts this key with the user 2's RSA public key and also encrypts the message with the AES secret key.
   (a) Prove that this AES key is too small to encrypt securely with RSA.
   (b) Provide an example of a public key that can be recovered by an attacker.
9. Consider again the combination of the two encryption methods discussed in the previous problem. What solutions would be feasible to overcome the mentioned vulnerability of this method?
10. In the Diffie-Hillman key-exchange protocol, prove that the two keys k_1 and k_2 are equal.
11. *Computer simulation project*. Write a computer program to simulate the operation of packet filtering in firewalls. Your program must include the following features.
    (a) The firewall sets up to allow in only HTTP requests to a specific server.
    (b) An outside intruder attempts to send packets with a forged internal address.
    (c) All packets from a given remote server must be kept out.

PART II

ADVANCED CONCEPTS

CHAPTER 11

Packet Queues and Delay Analysis

Queueing models offer qualitative insights into the performance of communication networks and quantitative estimations of average packet delay. In many networking instances, a single buffer forms a *queue* of packets. A single queue of packets is a notion of packet accumulation at a certain router or even at an entire network. In the context of data communication, a *queuing buffer* is a physical system that stores incoming packets, and a *server* can be viewed as a switch or a similar mechanism to process and route packets to the desired ports or destinations. A *queueing system* generally consists of a queueing buffer of various sizes and one or more identical servers. this chapter focuses on delay analysis of single queueing units and queueing networks, including feedback. The following topics are covered:

- *Little's theorem*
- *Birth-and-death process*
- *Queueing disciplines*
- *Markovian FIFO queueing systems*
- *Non-Markovian and self-similar models*
- *Network of queues and delay models*

The chapter starts by analyzing two basic theorems: Little's theorem and birth-and-death processes. Next, various scenarios of queueing disciplines are presented: finite versus infinite queueing capacity, one server versus several servers, and Markovian versus non-Markovian systems. Non-Markovian models are essential, as many network applications such as Ethernet, WWW, and multimedia traffic, cannot be modeled by Markovian patterns.

Networks of queues rely on two important and fundamental theorems: *Burke's theorem* and *Jackson's theorem*. Burke's theorem presents solutions to a network of several queues. Jackson's theorem is used when a packet visits a particular queue more than once. In such conditions, the network typically contains *loops* or *feedback*.

11.1 Little's Theorem

The fundamental concept of packet queueing is based on *Little's theorem*, or *Little's formula*. This formula states that for networks reaching steady state, the average number of packets in a system is equal to the product of the average arrival rate, λ, and the average time spent in the queueing system. For example, consider the communication system shown in Figure 11.1. Assume that the system starts with empty state at time $t = 0$. Let A_i and D_i be the ith arriving packet and ith departing packet, respectively, and let $A(t)$ and $D(t)$ be the total number of arriving packets and the total number of departing packets, respectively, up to a given time t.

Thus, the total time a packet i spends in the system is expressed by $T_i = D_i - A_i$, and the total number of packets in the system at time t is described by $K(t) = A(t) - D(t)$. The cumulative timing chart of packets in a first-come, first-served service discipline is shown in Figure 11.2. The total time that all packets spend in the system can be expressed by $\sum_{i=1}^{A(t)} T_i$. Thus, the average number of packets in the system, K_a, can be obtained by

$$K_a = \frac{1}{t} \sum_{i=1}^{A(t)} T_i. \tag{11.1}$$

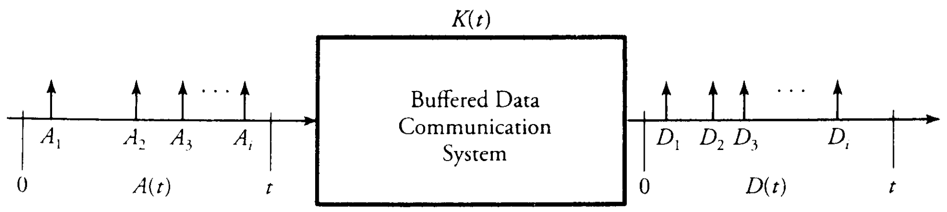


Figure 11.1 Overview of a buffered communication system with arriving and departing packets

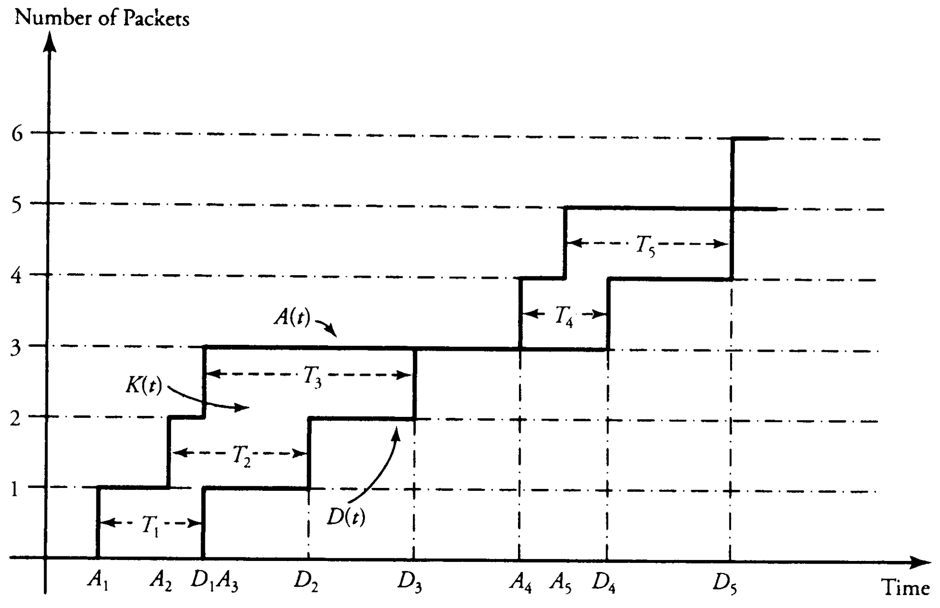


Figure 11.2 Accumulation of 5 packets in a queueing system

Similarly, the average time a packet spends in the system, T_a, can be represented by

$$T_a = \frac{1}{A(t)} \sum_{i=1}^{A(t)} T_i. \tag{11.2}$$

Also, the average number of arriving packets per second is given by λ as

$$\lambda = \frac{A(t)}{t}. \tag{11.3}$$

By combining Equations (11.1), (11.2), and (11.3), we obtain

$$K_a = \lambda T_a. \tag{11.4}$$

This important result leads to a final form: Little's formula. This formula makes the assumption that t is large enough and that K_a and T_a therefore converge to their expected values of corresponding random processes $E[K(t)]$ and $E[T]$, respectively. Thus:

$$E[K(t)] = \lambda E[T]. \tag{11.5}$$

Little's formula holds for most service disciplines with an arbitrary number of servers. Therefore, the assumption of first-come, first-served service discipline is not necessary.

Example. Consider a data-communication system with three transmission lines; packets arrive at three different nodes with arrival rates $\lambda_1 = 200$ packets/sec, $\lambda_2 = 300$ packets/sec, and $\lambda_3 = 10$ packets/sec, respectively. Assume that an average of 50,000 same-size packets float in this system. Find the average delay per packet in the system.

Solution. This delay is derived by

$$T_a = \frac{K_a}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{50,000}{200 + 300 + 10} = 98.4 \text{ sec.}$$

This is a simple application of Little's formula in a communication system.

11.2 Birth-and-Death Process

If a packet arrives at or leaves from a queueing node of a computer network, the state of the node's buffer changes, and thus the state of the queue changes as well. In such cases, as discussed in Appendix C, if the system can be expressed by a *Markov process*, the activity of the process—in terms of the number of packets—can be depicted by a state machine known as the *Markov chain*. A particular instance of a Markov chain is the *birth-and-death process*.

In a *birth-and-death process*, any given state i can connect only to state $i - 1$ with rate μ_i or to state $i + 1$ with rate λ_i, as shown in Figure 11.3. In general, if $A(t)$ and $D(t)$ are the total number of arriving packets and the total number of departing packets, respectively, up to given time t, the total number of packets in the system at time t is described by $K(t) = A(t) - D(t)$. With this analysis, $A(t)$ is the total number of *births*, and $D(t)$ is the total number of *deaths*. Therefore, $K(t)$ can be viewed as a birth-and-death process representing the cumulative number of packets in a first-come, first-served service discipline.

Let p_i be the probability that the chain is in state i. The balance equation for the steady-state probability at state 0 denoted by p_0 is related to the one for State 1 denoted by p_1 as

$$\lambda_0 p_0 = \mu_1 p_1. \tag{11.6}$$

Similarly, for state 1, the balance equation is

$$\lambda_0 p_0 + \mu_2 p_2 = \lambda_1 p_1 + \mu_1 p_1, \tag{11.7}$$

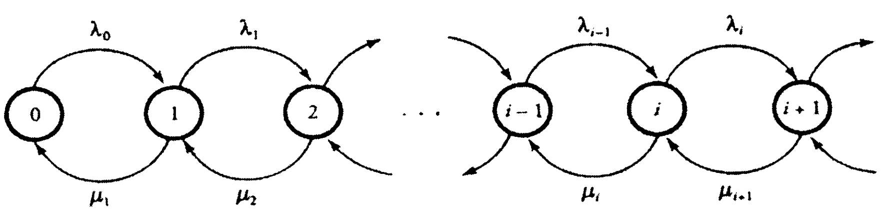


Figure 11.3 A general birth-and-death process and its transitions

and by using Equation (11.6), we derive the balance equation of state 1 as $\lambda_1 p_1 - \mu_2 p_2 = 0$. This development can continue up to state $i-1$, where the balance equation can be derived as

$$\lambda_{i-1} p_{i-1} - \mu_i p_i = 0. \tag{11.8}$$

We define $\rho_i = \frac{\lambda_{i-1}}{\mu_i}$ as *state i utilization*. Thus, we can express the balance Equation (11.8) by a generic form as

$$p_i = \rho_i p_{i-1}. \tag{11.9}$$

If we apply Equation (11.9) for every state, starting with $p_1 = \rho_1 p_0$ up to $p_i = \rho_i p_{i-1}$, and combine all the equations, we obtain a generic equation

$$p_i = (\rho_1 \rho_2 \cdots \rho_i) p_0. \tag{11.10}$$

Since the sum of all the probabilities is always equal to 1, as $\sum_{i=0}^{\infty} Pi = \sum_{i=0}^{\infty} (\rho_1 \rho_2 \ldots \rho_i p_0) = 1$, p_i in Equation (11.10) can also be obtained from

$$p_i = \frac{\rho_1 \rho_2 \ldots \rho_i}{\sum_{j=0}^{\infty} (\rho_1 \rho_2 \ldots \rho_i)}. \tag{11.11}$$

The significance of this result is that a state probability p_i is expressed in terms of state utilizations, $\rho_1, \rho_2, \ldots, \rho_1$ only.

11.3 Queueing Disciplines

Queueing systems are widely described by *Kendal's notations* as *A/B/m/K*, where A denotes the distribution of the interarrival times; *B*, the distribution of the service times; *m*, the number of servers; and *K*, the total capacity of the system. If a system reaches its full capacity, the arriving packet number $K + 1$ is blocked away. A and B are normally represented by the following symbols:

M	Exponential distribution
E_k	Erlang distribution with k phases
H_k	Hyperexponential distribution with k phases
D	Deterministic distribution
G	General distribution
GI	General distribution with independent interarrival times
$MMPP$	Markov-modulated Poisson process
$BMAP$	Batch Markovian arrival process

In a queueing system, packets arrive at the buffer and are stored there. If other packets are waiting for service in the queue, the incoming packet is stored as long as all other packets are ahead of it. Thus, a server state can be either *idle* or *busy*. When a currently in-service packet departs from the system, one of the stored packets is selected for service according to the *scheduling* discipline. A commonly used arriving process assumes that the sequence of interarrival times is *independent and identically distributed* (IID).

Queueing systems can be classified as follows:

- *First come, first served* (FCFS). Packets are served in the order they arrive.
- *Last come, first served* (LCFS). Packets are served in reverse order of their arrival.
- *Random service*. Packets are selected at random for service.
- *Priority service*. The selection of a packet in the queue depends on priorities that are locally or permanently assigned to the packet.
- *Preemption*. The packet currently being serviced can be interrupted and preempted if a packet in the queue has a higher priority.
- *Round robin*. If the time allowed for the service of a packet is through and the process is not finished, the packet returns to the queue, and the action is repeated until the job service is completed.
- *Service sharing*. Servers can share their power with one another to provide services to the packets requiring more processing time.

This chapter focuses on FIFO models to show how the fundamental queueing in computer networks works. Chapter 12 looks at the applications of more advanced queuing models, such as priority, preemption, and round-robin queues.

11.4 Markovian FIFO Queueing Systems

In *Markovian* queueing systems, both arrival and service behaviors are based on Markovian-model queueing disciplines: $M/M/1$, $M/M/1/b$, $M/M/a$, $M/M/a/a$, and $M/M/\infty$.

11.4.1 *M/M/1* Queueing Systems

Figure 11.4 shows a simple queueing system model. Packets arrive according to a Poisson model at rate λ so the interarrival times are IID exponential random variables with mean $1/\lambda$. When the server is prepared for service, packets from the queue enter the server. Consider the situation in which the service times are IID exponential random variables with mean $1/\mu$ and the interarrival and service times are independent. With the $M/M/1$ model, it is in fact assumed that the system can accommodate an unlimited number of packets.

Packet Arrival and Service Model

In an $M/M/1$ model, the *packet arrival* distribution is Poisson (see Appendix C) with rate λ, or the interarrival time distribution is exponential with mean time $1/\lambda$. Similarly, the service rate distribution is Poisson with rate μ, or the service time distribution is exponential with mean time $1/\mu$. Note that the interarrival time and the service time are independent. Figure 11.5 shows an example in which five packets, A_1 through A_5, arrive in random and require service times S_1 through S_5, respectively. In this situation,

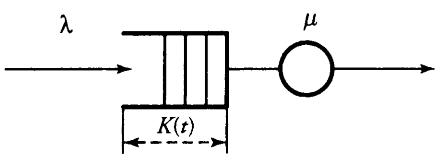


Figure 11.4 A simple queue/server model

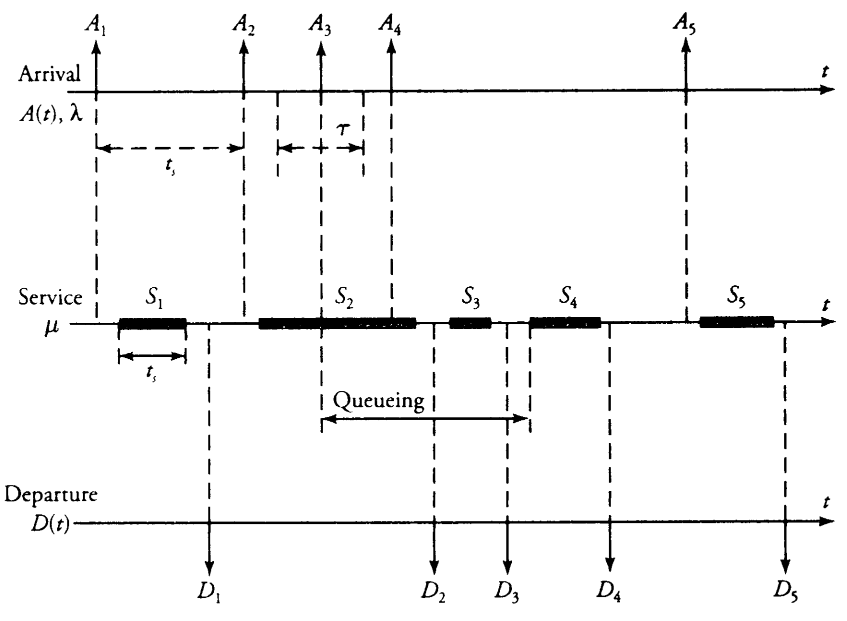


Figure 11.5 Packet arrival, service, and departure

queueing occurs while packets A_3 and A_4 wait in the buffer until the server becomes available. The probability that one packet arrives in an interval τ but no packet departs from the system is obtained using the Poisson formula:

$$
\begin{aligned}
P[1 \text{ packet arrives, } 0 \text{ packets depart}] &= \frac{\lambda\tau e^{-\lambda\tau}}{1!} \\
&= \lambda\tau\left(1 - \frac{\lambda\tau}{1!} + \frac{(\lambda\tau)^2}{2!} - \cdots\right) \\
&= \lambda\tau + \left(-\frac{(\lambda\tau)^2}{1!} + \frac{(\lambda\tau)^3}{2!} - \cdots\right) \\
&\approx \lambda\tau. \qquad (11.12)
\end{aligned}
$$

Equation (11.12) assumes that interval τ becomes very small and that therefore, all terms except the first one can be ignored. The same analysis can be derived for the case in which no packets arrive in an interval τ, but one packet departs from the

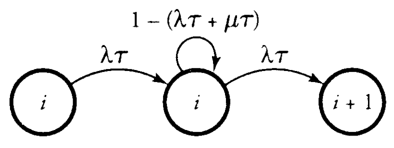


Figure 11.6 Packet arrival and departure

system. Because one packet is serviced in an interval τ, we can use rate μ instead of λ in Equation (11.12). Thus:

$$\begin{aligned}
\text{P[0 packets arrive, 1 packet departs]} &= \mu\tau\frac{e^{-\mu\tau}}{1!} \\
&= \mu\tau(1 - \frac{\mu\tau}{1!} + \frac{(\mu\tau)^2}{2!} - \cdots) \\
&= \mu\tau + (-\frac{(\mu\tau)^2}{1!} + \frac{(\mu\tau)^3}{2!} - \cdots) \\
&\approx \mu\tau. \qquad (11.13)
\end{aligned}$$

The last situation occurs when no change in the number of packets in the system is made. This means that there can be one arriving packet and one departing packet, or there can be no arriving packet and no departing packet. In either case, the probability can be derived by using Equations (11.12) and (11.13):

$$\text{P[no changes in the number of packets]} \approx 1 - (\lambda\tau + \mu\tau). \qquad (11.14)$$

This analysis of the process is illustrated in Figure 11.6. In this process, a chain starts from a given state i, implying i packets in the system.

Number of Packets in the System

Properties obtained from Equations (11.12), (11.13), and (11.14) lead to the derivation of the number of packets in an $M/M/1$ system: $K(t)$. The fact is that $K(t)$ follows a birth-and-death process, and its Markov chain follows exactly as the generic one shown in Figure 11.3 but with equal rates as modified in Figure 11.7. The main property of the exponential random variable implies that the interarrival time is independent of the present and past status of $K(t)$. Thus, if the system has $K(t) > 0$ packets, the interdeparture time distribution is also an exponential random variable. This also implies that the past history of the system is irrelevant for the probabilities of future

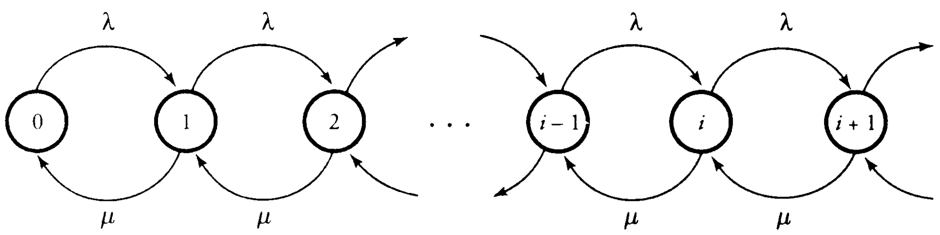


Figure 11.7 $M/M/1$ queueing behavior and its transitions

states, and thus the process follows the property of a Markov chain. Note that $M/M/1$ is a simple case of a birth-death process with:

$$\lambda_0 = \lambda_1 = \lambda_2 = \cdots = \lambda \tag{11.15}$$

and

$$\mu_0 = \mu_1 = \mu_2 = \cdots = \mu. \tag{11.16}$$

We define $\rho = \frac{\lambda}{\mu}$ as the system *utilization.* The preceding Markov chain has the state-transition rate diagram shown in Figure 11.7, and its global balance equations for the steady-state probabilities can be set by using the birth-death process results simplified as

$$p_i = (\rho\rho \cdots \rho)p_0 = \rho^i p_0. \tag{11.17}$$

Since the sum of all probabilities is always 1, we can write

$$\sum_{i=0}^{\infty} p_i = \sum_{i=0}^{\infty} \rho^i p_0 = 1. \tag{11.18}$$

A stable situation exists when the arrival rate is less than the service rate, or $\rho = \lambda/\mu < 1$, in which case $K(t)$ does not increase in value without bound, and therefore $\sum_{i=0}^{\infty} \rho^i \to \frac{1}{1-\rho}$. This fact simplifies Equation (11.17) for the derivation of the number of packets in the system:

$$p_i = P[K(t) = i] = (1 - \rho)\rho^i \quad i = 1, 2, \ldots. \tag{11.19}$$

The condition $\rho = \lambda/\mu < 1$ must be met if the system is designed to be stable; otherwise, packets arrive at the system more quickly than they can be processed. It is apparent that if the packet-arrival rate exceeds the processing-capacity rate of servers, the number of packets in the queue grows without bound, leading to an unstable situation. Note that the distribution of $K(t)$ follows a *geometric distribution*.

Mean Delay and Queue Length

The result derived from Equation (11.19) can be used to calculate the mean delay a packet spends in a queue or the entire system and the mean length of a queue. The average number of packets in the system is, in fact, the expected value of $K(t)$:

$$\begin{aligned} E[K(t)] &= \sum_{i=0}^{\infty} iP[K(t)=i] = \sum_{i=0}^{\infty} i(1-\rho)\rho^i = (1-\rho)\sum_{i=0}^{\infty} i\rho^i \\ &= \frac{\rho}{1-\rho}. \end{aligned} \tag{11.20}$$

Little's formula can be used to derive the mean packet delay in the system:

$$E[T] = \frac{1}{\lambda}E[K(t)] = \frac{1}{\lambda}\frac{\rho}{1-\rho} = \frac{1}{\mu-\lambda}. \tag{11.21}$$

We can now show the mean waiting time of a packet in the queue, $E[T_q]$, in terms of $E[T]$ and the mean processing time of a packet in the server, $E[T_s]$:

$$E[T_q] = E[T] - E[T_s]. \tag{11.22}$$

The mean processing time of a packet in the server, $E[T_s]$, is in fact $\frac{1}{\mu}$; therefore:

$$\begin{aligned} E[T_q] &= \frac{1}{\lambda}\frac{\rho}{1-\rho} - \frac{1}{\mu} \\ &= \frac{\lambda}{\mu}\frac{1}{\mu-\lambda}. \end{aligned} \tag{11.23}$$

Example. Use Little's formula to find the mean number of packets in the queue.

Solution. Little's formula can be applied in most stochastic systems, such as a queue defined solely as a system. If $E[K_q(t)]$ is the mean number of packets in the queue:

$$E[K_q(t)] = \lambda E[T_q] = \frac{\rho^2}{1-\rho}.$$

Interestingly, the mean number of packets is a function only of utilization. In other words, the value of utilization in any single-queue system determines the average number of packets in the system.

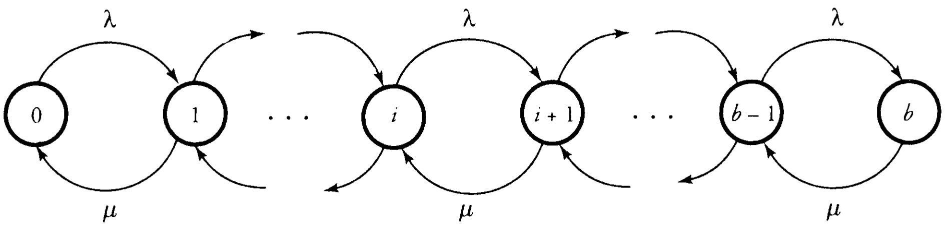


Figure 11.8 State-transition diagram for $M/M/1/b$ system

11.4.2 Systems with Limited Queueing Space: $M/M/1/b$

In a queue with exponential interarrival times and exponential service times, a single server and a maximum of b packets can be held in the system. Once such a system exceeds its capacity, arriving packet numbered $b + 1$ is turned away from the system, since there is no room for it. Figure 11.8 shows the state-transition diagram for the $M/M/1/b$ system. This diagram is a Markov chain and is very similar to the one illustrated in Figure 11.7, but the chain for the $M/M/1/b$ system stops at the maximum capacity of b. Thus, the global balance equations use the same results presented in Equation (11.18), where the sum of all probabilities is always 1 and thus

$$\sum_{i=0}^{b} p_i = \sum_{i=0}^{b} \rho^i p_0 = 1. \tag{11.24}$$

A stable situation exists when the arrival rate is less than the service rate, or $\rho = \lambda/\mu < 1$, in which case $K(t)$ does not increase in value without bound and therefore $\sum_{i=0}^{b} \rho^i \rightarrow \frac{1-\rho^{b+1}}{1-\rho}$. This fact simplifies Equation (11.17) for deriving the number of packets in the system, which is a continuous-time Markov chain taking on values from the set $\{0, 1, \cdots, b\}$:

$$p_i = P[K(t) = i] = \begin{cases} \rho^i \frac{1-\rho}{1-\rho^{b+1}} & \text{if } \rho \neq 1 \text{ and } i = 1, 2, \cdots b \\ \frac{1}{1+b} & \text{if } \rho = 1 \end{cases} \tag{11.25}$$

Note that the number of packets in the system is derived in two parts, depending on whether the utilization is maximized ($\rho = 1$) or not.

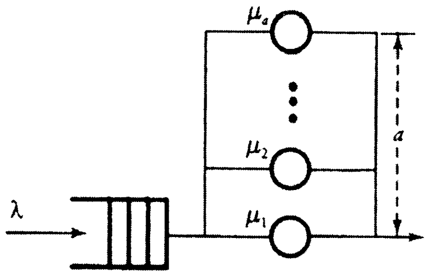


Figure 11.9 Queueing model of *M/M/a* systems

Mean Number of Packets

The mean number of packets in the system, *E[K(t)]*, can be expressed by Formula C.20 in Appendix C:

$$\mathrm{E}[K(t)] = \sum_{i=0}^{b} iP[K(t)=i] = \begin{cases} \frac{\rho}{1-\rho} - \frac{(b+1)\rho^{b+1}}{1-\rho^{b+1}} & \text{if } \rho \neq 1 \\ \frac{b}{2} & \text{if } \rho = 1 \end{cases}. \tag{11.26}$$

Clearly, results of the mean number of packets in the *M/M/*1 and *M/M/*1/*b* systems are fundamentally different, owing to the difference in capacities available in these two systems.

11.4.3 *M/M/a* Queueing Systems

Some stochastic systems are modeled by *M/M/a*, where *a* servers instead of one can be used to enhance system performance, as shown in Figure 11.9. The service rate in each server may be different from or equal to that in the others, as rates μ_1 through μ_a are indicated as service rates of the corresponding servers. The advantage of this discipline over systems with one server is its parallel-serving feature. A packet can be partitioned into $i < a$ different tasks and served in parallel in i different servers concurrently.

The resulting system can be modeled by a continuous-time Markov chain, as shown in Figure 11.10. The maximum number of servers is denoted by a. As long as the number of packets in the queue is less than a, the service rate is proportionally changing ($i\mu$) with the number of waiting packets while the service rate remains constant. However, when the number of packets in the queue reaches a, the arrival rate stays constant at its maximum of $a\mu$.

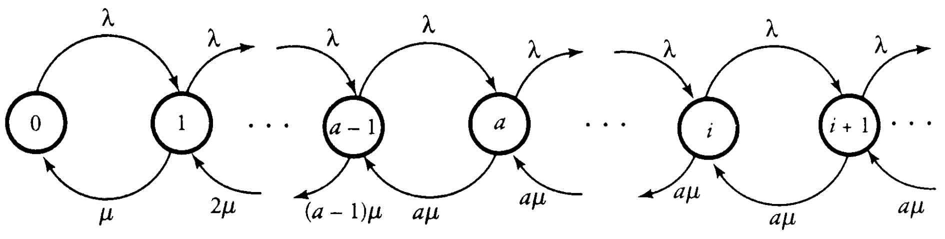


Figure 11.10 State-transition diagram for $M/M/a$ systems

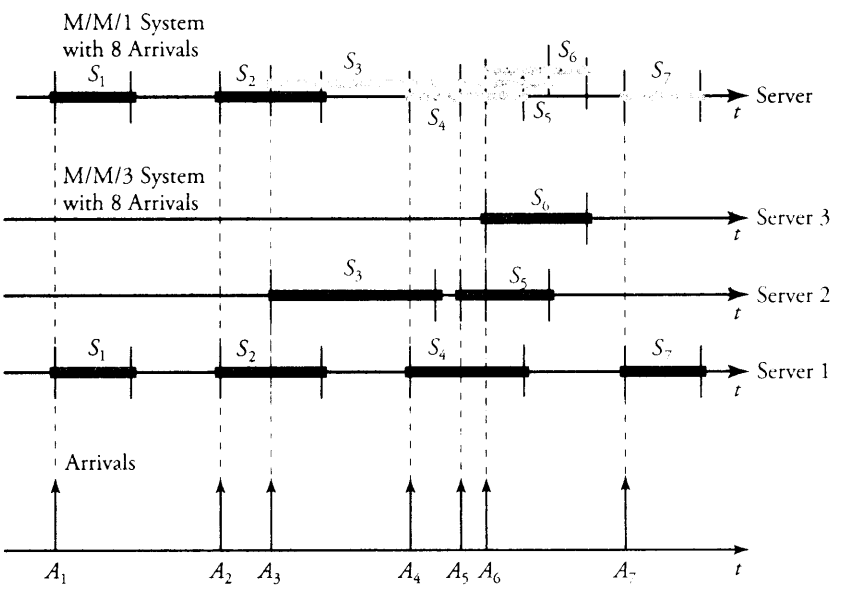


Figure 11.11 Comparison of $M/M/1$ and $M/M/3$ systems, each with seven packets

Example. Figure 11.11 shows a statistical comparison between $M/M/1$ and $M/M/3$ systems, each with seven packets. In the $M/M/1$ system, the third packet, S_3, starts to accumulate in the queue. The queueing trend continues, as shown by gray, if the service rate stays low. On the contrary, in the $M/M/3$ system, packets are distributed over the three servers on arrival, and queueing has not happened.

Balance Equations for $i \leq a$

The steady-state probabilities for the *M/M/a* system can be obtained by using the birth-and-death process. Balance equations for i servers being busy when $i \leq a$ begin with p_i:

$$p_i = \left(\frac{\lambda}{i\mu} p_{i-1}\right) \quad i = 1, \ldots, a. \tag{11.27}$$

Recall that $\rho_i = \frac{\lambda}{i\mu}$ is the utilization and that $p_i = (\rho_1 \rho_2 \ldots \rho_{i-1} \rho_i)\, p_0$. Thus, p_i can be rewritten as

$$p_i = \left(\frac{\lambda}{\mu}\frac{\lambda}{2\mu} \cdots \frac{\lambda}{(i-1)\mu}\frac{\lambda}{i\mu}\right) p_0 = \frac{\left(\frac{\lambda}{\mu}\right)^i}{i!} p_0 = \left(\frac{\rho^i_1}{i!}\right) p_0. \tag{11.28}$$

It is easy to realize the probability that all servers are in use. Substitute $i = a$ in Equation (11.28):

$$p_a = \left(\frac{\rho_1^a}{a!}\right) p_0 = \frac{\left(\frac{\lambda}{\mu}\right)^a}{a!} p. \tag{11.29}$$

Balance Equations for $i \geq a$

To determine the balance equations for $i \geq a$, we again use the birth-and-death process. The second part of the transition diagram, starting from state a, is formulated over states $i = a, a + 1, a + 2, \ldots$. As a result, the birth-and-death process formula is set up by

$$p_i = \rho^{i-a}\, p_a. \tag{11.30}$$

where $\rho = \frac{\lambda}{a\mu}$. In this equation, p_a can be substituted by the value obtained last from the case of $i \leq a$, presented in Equation (11.29). Therefore, by replacing p_a with $\frac{\rho_1^a}{a!} p_a$, we obtain

$$p_i = \frac{\rho^{i-a} \rho_1^a}{a!} p_0. \tag{11.31}$$

Equations (11.28) and (11.31) together cover all the cases in the state diagram. Since the sum of all probabilities is always 1, $\sum_{i=0}^{\infty} p_i = 1$:

$$\sum_{i=0}^{a-1} \frac{\rho_1^i}{i!} p_0 + \frac{\rho_1^a}{a!} \sum_{i=a}^{\infty} \rho^{i-a} p_0 = 1. \tag{11.32}$$

To have a stable system, we want $\rho < 1$, which causes $\sum_{i=a}^{\infty} \rho^{i-a}$ to converge to $\frac{1}{1-\rho}$. With this simplification, we can compute the probability of the first state from Equation 11.32:

$$p_0 = \frac{1}{\sum_{i=0}^{a-1} \frac{\rho_1^i}{i!} + \frac{\rho_1^a}{a!} \frac{1}{1-\rho}}. \tag{11.33}$$

Similar to previous cases, knowing p_0 is essential, as we can derive p_i from Equation (11.31).

In an $M/M/a$ system, we can use the *Erlang-C formula* to compute the blocking probability. This formula calculates the probability that an arriving packet finds all servers busy. This probability is the same as the probability that the waiting time for the packet in the queue is greater than zero, or the probability that the number of packets in the queueing system is greater than or equal to a, $P[K(t) \geq a]$:

$$\text{Erlang-C Blocking Prob.: } P[K(t) \geq a] = \sum_{i=a}^{\infty} p_i = \frac{p_a}{1-\rho}. \tag{11.34}$$

As with the $M/M/1$ queueing system, we can now estimate all the parameters for the $M/M/a$ discipline. The mean number of packets in the queue is

$$\begin{aligned} E[K_q(t)] &= \sum_{i=a}^{\infty}(i-a)p_i = \sum_{i=a}^{\infty}(i-a)\rho^{i-a}p_a = p_a \sum_{i-a=0}^{\infty}(i-a)\rho^{i-a} \\ &= \frac{p_a\rho}{(1-\rho)^2}. \end{aligned} \tag{11.35}$$

Accordingly, the mean packet delay in the queue is

$$E[T_q] = \frac{1}{\lambda}E[K_q(t)] = \frac{p_a}{a\mu(1-\rho)^2}. \tag{11.36}$$

Obviously, the total mean packet delay in a system is

$$E[T] = E[T_q] + E[T_s] = \frac{p_a}{a\mu(1-\rho)^2} + \frac{1}{\mu}. \tag{11.37}$$

The mean number of packets in the system is obtained again from Little's formula:

$$\begin{aligned} E[K(t)] &= \lambda E[T] = \lambda\left(\frac{p_a}{a\mu(1-\rho)^2} + \frac{1}{\mu}\right) \\ &= \frac{p_a\rho}{(1-\rho)^2} + \rho_1. \end{aligned} \tag{11.38}$$

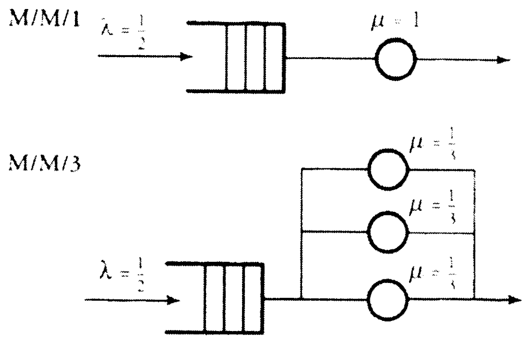


Figure 11.12 Two queueing disciplines with equal service powers

Example. A communication node receives packets at $\lambda = 1/2$ packet per nanosecond, and the switching server processes packets at the rate of one packet per nanosecond. Calculate the queueing delay and the total node's delay, and then compare the results for using queueing disciplines *M/M/1* and *M/M/3* with equal service powers as shown in Figure 11.12.

Solution. With an *M/M/1* discipline, $\rho = \lambda/\mu = 0.5$. So the mean queueing delay is

$$E[T_q] = \frac{\rho/\mu}{1-\rho} = 1 \text{ nsec.} \tag{11.39}$$

and the total node's delay is

$$E[T] = \frac{1/\mu}{1-\rho} = 2\,\text{nsec.} \tag{11.39}$$

For the *M/M/3* discipline, $a = 3$ servers are used; thus, $\rho_1 = \lambda/\mu = 1.5$, and $\rho = \lambda/a\mu = 0.5$. Thus:

$$P_0 = \frac{1}{\sum_{i=0}^{2} \frac{\rho_1^i}{i!} + \frac{\rho_1^3}{3!}\frac{1}{1-\rho}}. \tag{11.41}$$

Since $p_a = \frac{\rho_1^a}{a!} p_0 = 0.095$, the queueing delay is

$$E[T_q] = p_a \frac{1}{a\mu(1-\rho)^2} = 0.38, \tag{11.42}$$

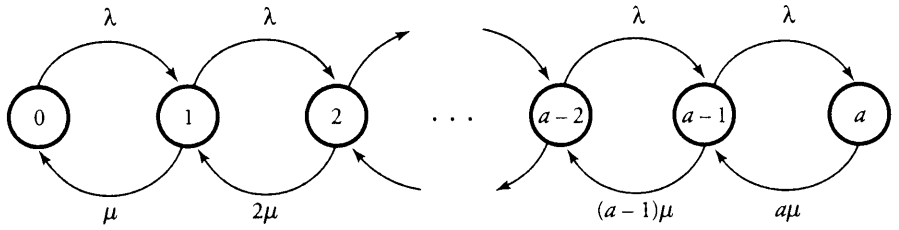


Figure 11.13 State-transition diagram for $M/M/a/a$ systems

and the total mean node's delay is

$$E[T] = E[T_q] + E[T_s] = 0.38 + \frac{1}{\mu} = 3.38. \qquad (11.43)$$

This useful example expresses an important conclusion: *Increasing the number of servers lowers the queueing delay but increases the total system delay.* In summary, compared with the $M/M/a$ system ($a > 1$), the $M/M/1$ system has a smaller total delay but a larger queueing delay. However, note that the preceding statement should be made with an important condition; that is, the total service rates of a servers in the $M/M/a$ model must be equal to the service rate of the $M/M/1$ model.

11.4.4 Models for Delay-Sensitive Traffic: *M/M/a/a*

Some network nodes can be built with no buffering elements. One application of such systems is for cases in which *delay-sensitive traffic*, such as voice, needs a queueless service. These types of nodes can be shown by queueless models. One example of such models is the $M/M/a/a$ queueing system, which has a servers but no queueing line. If all servers are busy, an arriving packet is turned away, resulting in the transition-rate diagram shown in Figure 11.13. This transition rate partially resembles that for an $M/M/a$ system. Thus, we use the previously obtained equation for the probability that the system is in state $i \in \{0, \cdots, a\}$:

$$p_i = \left(\frac{\rho_1^i}{i!}\right) p_0. \qquad (11.44)$$

Combining Equation 11.44 and the fact that $\sum_{i=0}^{a} p_i = 1$ gives us

$$p_0 = \frac{1}{\sum_{i=0}^{a} \frac{\rho_1^i}{i!}}. \qquad (11.45)$$

Erlang-B Blocking Probability

In an $M/M/a/a$ system, we can compute the blocking probability, this time using the *Erlang-B formula*. This formula calculates the probability that an arriving packet finds all servers busy but no waiting line for packets. The probability that all servers of the system are busy is referred to as the *Erlang-B formula*, using Equation (11.44) with $i = a$:

$$\text{Erlang-B Blocking Prob.: } P[K(t) = a] = p_a = \frac{\rho_1^a}{a!} p_0$$

$$= \frac{\rho_1^a}{a!\left(\sum_{i=0}^{a} \frac{\rho_1^i}{i!}\right)}. \qquad (11.46)$$

When all servers are busy, an incoming packet is turned away. Let λ, λ_p, and $\lambda_b = p_a\lambda$ represent the arrival rate, the packet-passing rate, and the blocking rate, respectively. Therefore, $\lambda = \lambda_p + \lambda_b$. The packet-passing rate can then be rewritten as $\lambda_p = \lambda(1 - p_a)$. In such systems, it would be interesting to find the average number of packets in the system, since no queueing line exists. From Little's formula:

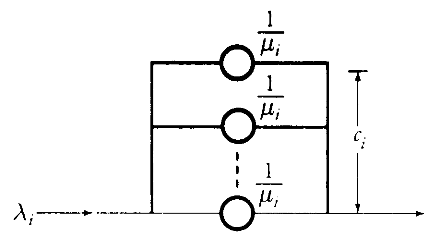
$$E[K(t)] = \lambda_p E[T_s] = \lambda(1 - p_a)\frac{1}{\mu}. \qquad (11.47)$$

Example. When a mobile user in a wireless data network moves to a new region, a handoff process between the two regions needs to take place. If all the new region's channels are used, the handoff call is terminated or blocked. For real-time communications, this scenario is modeled by an $M/M/a/a$ system, where $a = c_i$ is the number of handoff servers of traffic type i. Assume that c_i=10, 50, and 100; mean service time $1/\mu_i = 30$ msec; and handoff request rate $\lambda_i = 0 \cdots 10, 000$. Sketch a set of plots showing the performance of the handoff process.

Solution. The calculation effort is left to readers. Instead, the results of the computations in Figure 11.14 are given. The plots in this figure depict the results of an $M/M/a/a$ model for the wireless handoff. Equation (11.46) can be used to compute the handoff blocking probability.

11.4.5 $M/M/\infty$ Queueing Systems

In many high-speed network nodes, higher cost can be tolerated in exchange for better communication quality. One method of improving the quality of handling packets is to consider a large number of servers at a node. If we let a approach a real large number approximated to infinity, an $M/M/a/a$ system becomes an $M/M/\infty$ system. As a



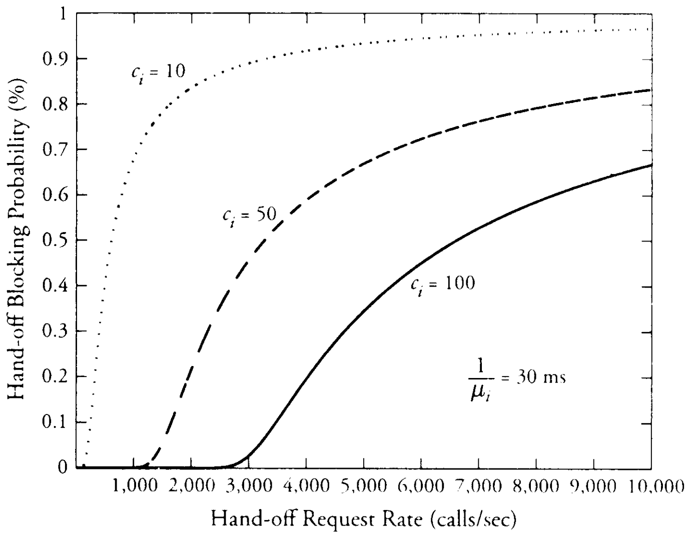


Figure 11.14 A wireless data network model as an $M/M/a/a$ system with c_i channel servers

result, the transition-rate diagram can continue to ∞, as shown in Figure 11.15. To find steady-state probabilities, p_i, for this queueing system, we use Equation (11.46) of the $M/M/a/a$ system and let a be a number i varying up to ∞:

$$p_i = \frac{\rho_1^i}{i!\left(\sum_{j=0}^{i} \frac{\rho_1^j}{j!}\right)}. \tag{11.48}$$

In the denominator of this equation, $\sum_{j=0}^{i} \frac{\rho_1^j}{j!}$ tends to converge to e^{ρ_1} when i approaches infinity:

$$p_i = \frac{\rho_1^i}{i!e^{\rho_1}}. \tag{11.49}$$

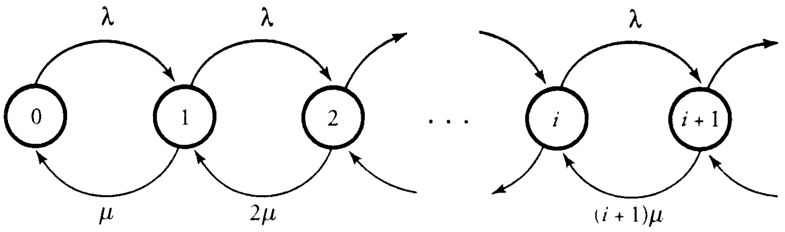


Figure 11.15 State-transition diagram for $M/M/\infty$ system

11.5 Non-Markovian and Self-Similar Models

Another class of queueing systems is either completely or partially *non-Markovian.* Partial non-Markovian means that either arrival or service behavior is not based on Markovian models. We first consider a single server in which packets arrive according to a Poisson process, but the service-time distribution is *non-Poisson* or *non-Markovian* as *general distribution.* One example of a non-Markovian system is $M/G/1$ queues with priorities. Another is $M/D/1$, where the distribution of service time is deterministic, not random. Yet another example is reservation systems in which part of the service time is occupied with sending packets—serving packets—and part with sending control information or making reservations for sending the packets.

11.5.1 Pollaczek-Khinchin Formula and *M/G/1*

Let us consider a single-server scenario in which packets arrive according to a Poisson process with rate λ, but packet service times have a *general distribution.* This means that the distribution of service times cannot necessarily be exponential, as shown in Figure 11.16. Suppose that packets are served in the order they are received, with T_{si} the service time of the ith packet. In our analysis, we assume that random variables T_{s1}, T_{s2}, $\cdots$ are identically distributed, mutually independent, and independent of the interarrival times. Let T_{qi} be the queueing time of the ith packet. Then:

$$T_{qi} = \pi_i + \sum_{j=i-K_{qi}}^{i-1} T_{sj}, \tag{11.50}$$

where K_{qi} is the number of packets found by the ith packet waiting in the queue on arrival, and π_i is the partial service time seen by the ith packet if a given packet j is already being served when packet i arrives. If no packet is in service, π_i is zero. In order

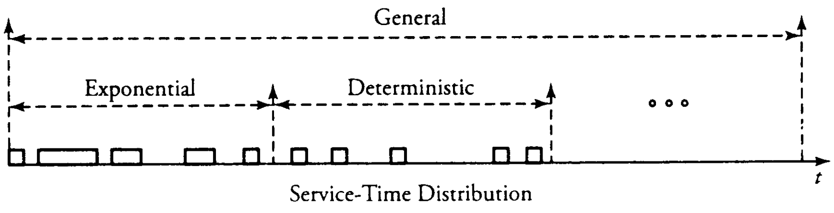


Figure 11.16 A general distribution of service times

to understand the averages of the mentioned times, we take expectations of two sides in Equation (11.50):

$$E[T_{qi}] = E[\pi_i] + E[\sum_{j=i-N_i}^{i-1} T_{sj}] = E[\pi_i] + E[\sum_{j=i-K_{qi}}^{i-1} E[T_{sj}|K_{qi}]]. \quad (11.51)$$

With the independence induction of the random variables K_{qi} and T_{qj}, we have

$$E[T_{qi}] = E[\pi_i] + E[T_{si}]E[K_{qi}]. \quad (11.52)$$

It is clear that we can apply the average service time as $E[T_{si}] = \frac{1}{\mu}$ in Equation (11.52). Also, by forcing Little's formula, where $E[K_{qi}] = \lambda E[T_{qi}]$, Equation (11.52) becomes

$$E[T_{qi}] = E[\pi_i] + \frac{1}{\mu}\lambda E[T_{qi}] = E[\pi_i] + \rho E[T_{qi}], \quad (11.53)$$

where $\rho = \lambda/\mu$ is the utilization. Thus,

$$E[T_{qi}] = \frac{E[\pi_i]}{1-\rho}. \quad (11.54)$$

Figure 11.17 shows an example of mean π_i representing the remaining time for the completion of an in-service packet as a function of time. If a new service of T_{si} seconds begins, π_i starts at value T_{si} and decays linearly at 45 degrees for T_{si} seconds. Thus, the mean π_i in the interval $[0, t]$ with n triangles is equal to the area of all triangles $(\sum_{i=1}^{n} \frac{1}{2}T_{si}^2)$ divided by total time (t):

$$E[\pi_i] = \frac{1}{t}\sum_{i=1}^{n} \frac{1}{2}T_{si}^2 = \frac{1}{2}\frac{n}{t}\frac{\sum_{i=1}^{n} T_{si}^2}{n}. \quad (11.55)$$

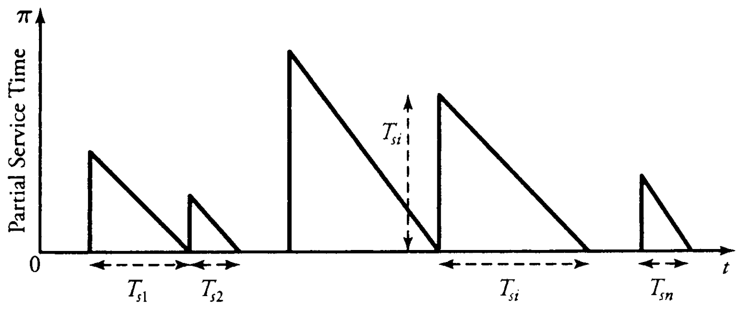


Figure 11.17 Mean partial service time

If n and t are large enough, $\frac{n}{t}$ becomes λ, and $\sum_{i=1}^{n} T_{si}^2/n$ becomes $E[T_{si}^2]$, the second moment of service time. Then, Equation (11.55) can be rewritten as

$$E[\pi_i] = \frac{1}{2}\lambda E[T_{si}^2]. \tag{11.56}$$

Combining Equations (11.55) and (11.54) forms the *Pollaczek-Khinchin* (P-K) formula:

$$E[T_{qi}] = \frac{\lambda E[T_{si}^2]}{2(1-\rho)}. \tag{11.57}$$

Clearly, the total queueing time and service time is $E[T_{qi}] + E[T_{si}]$. By inducing Little's formula, the expected number of packets in the queue, $E[K_{qi}]$, and the expected number of packets in the system, $E[K_i]$, we get

$$E[K_{qi}] = \frac{\lambda^2 E[T_{si}^2]}{2(1-\rho)} \tag{11.58}$$

and

$$E[K_i] = \rho + \frac{\lambda^2 E[T_{si}^2]}{2(1-\rho)}. \tag{11.59}$$

These results of the Pollaczek-Khinchin theorem express the behavior of a general distribution of service time modeled as $M/G/1$. For example, if an $M/M/1$ system needs to be expressed by these results, and since service times are exponentially distributed, we have $E[T_{si}^2] = 2/\mu^2$, and the Pollaczek-Khinchin formula reduces to

$$E[T_{qi}] = \frac{\rho}{\mu(1-\rho)}. \tag{11.60}$$

Note that the queueing metrics as the expected value of queuing delay obtained in Equation (11.60) is still a function of the system utilization. However, the notion of service rate μ has shown up in the equation and is a natural behavior of the non-Markovian service model.

11.5.2 *M/D/1* Models

The distribution of service time in a computer network is deterministic. In the *asynchronous transfer mode*, discussed in Chapter 2, packets are equal sized, and may be so ATM fits well fits in the $M/D/1$ model. When service times are identical for all packets, we have $E[T_{si}^2] = 1/\mu^2$. Then:

$$E[T_{qi}] = \frac{\rho}{2\mu(1-\rho)}. \qquad (11.61)$$

Note that the value of $E[T_{qi}]$ for an $M/D/1$ queue is the lower bound to the corresponding value for an $M/G/1$ model of the same λ and μ. This is true because the $M/D/1$ case yields the minimum possible value of $E[T_{si}^2]$. It is also worth noting that $E[T_{qi}]$ in an $M/M/1$ queue is twice as much as the one in an $M/D/1$ queue. The reason is that the mean service time is the same in the two cases, and for small ρs, most of the waiting occurs in the service unit, whereas for large ρs, most of the waiting occurs within the queue.

11.5.3 Self-Similarity and Batch-Arrival Models

The case of video streaming traffic, which obeys a model in the form of ***batch arrival***, not Poisson, is analyzed in Chapter 18, so our analysis on non-Poisson arrival models is presented there. Here, we give some descriptive and intuitive examples of such models, which give rise to *self-similar* traffic.

Non-Markovian arrival models, or basically ***non-Poisson arrival models***, have a number of important networking applications. The notion of self-similarity, for example, applies to WAN and LAN traffic. One example of such arrival situations is self-similarity traffic owing to the World Wide Web (WWW). The self-similarity in such traffic can be expressed based on the underlying distributions of WWW document sizes, the effects of caching and user preference in file transfer, and even user think time. Another example of self-similar traffic is video streaming mixed up with regular random traffic backgrounds. In such a case, batches of video packets are generated and are directed into a pool of mixed traffic in the Internet.

Traffic patterns indicate significant burstiness, or variations on a wide range of time scales. Bursty traffic can be described statistically by using *self-similarity patterns*. In a self-similar process, the distribution of a packet, frame, or object remains unchanged when viewed at varying scales. This type of traffic pattern has observable bursts and thus exhibits long-range dependence. The dependency means that all values at all times are typically correlated with the ones at all future instants. The long-range dependence in network traffic shows that packet loss and delay behavior are different from the ones in traditional network models using Poisson models.

11.6 Networks of Queues

Networks of queueing nodes can be modeled and their behavior, including feedback, studied. Two important and fundamental theorems are *Burke's theorem* and *Jackson's theorem*. Burke's theorem presents solutions to a network of several queues. Jackson's theorem is relevant when a packet visits a particular queue more than once.

11.6.1 Burke's Theorem

When a number of queueing nodes are connected, they all may be either in series or in parallel. In such circumstances, the entire system requires a careful and justified procedure for modeling. *Burke's theorem* presents solutions to a network of m queues: m queueing nodes in series and m queueing nodes in parallel.

Burke's Theorem on Cascaded Nodes

One common case is cascaded queueing nodes, as shown in Figure 11.18. In this network, the departure of a packet from a queueing node i is the arrival for node $i + 1$. The utilization for any node i is $\rho_i = \lambda/\mu_i$, where λ and μ_i are the arrival rate and service rate of that node, respectively.

The packet-movement activity of this network is modeled by an m-dimensional Markov chain. For simplicity, consider a two-node network of Nodes 1 and 2. Figure 11.19 shows a two-dimensional Markov chain for this simplified network. The chain starts at state 00. When the first packet arrives at rate λ, the chain goes to state 01. At this point, the chain can return to state 10 if it gets serviced at rate μ_1, since the packet is now in node 2. If the packet is sent out of the second node at service rate μ_2, the chain returns to state 00, showing the empty status of the system. The Markov chain can be interpreted similarly for a larger number of packets.

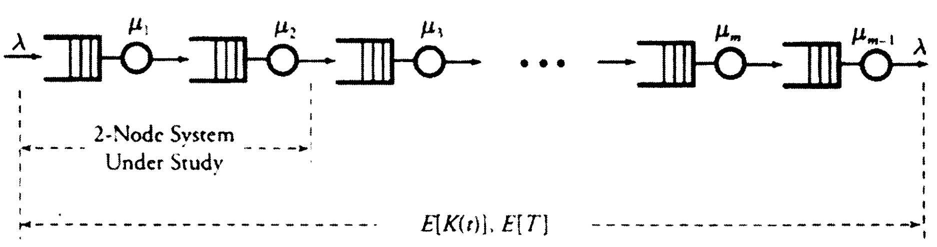


Figure 11.18 Burke's theorem applied on *m* cascaded queues

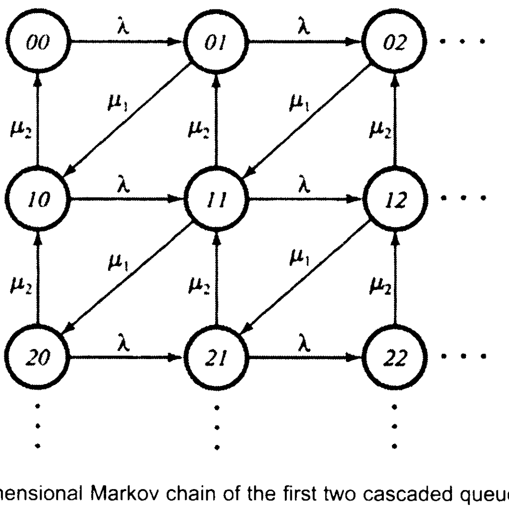


Figure 11.19 Two-dimensional Markov chain of the first two cascaded queues in Figure 11.18

In our system with *m* nodes, assume an *M/M/1* discipline for every node; the probability that *k* packets are in the system, $P[K(t) = k]$, is:

$$\begin{aligned} P[K(t)=k] &= P[K_1(t)=k_1]\times P[K_2(t)=k_2]\times\cdots\times P[K_m(t)=k_m] \\ &= \prod_{i=1}^{m}(1-\rho_i)\rho_i^{k_i}. \end{aligned} \tag{11.62}$$

The expected number of packets in each node, $E[K_i(t)]$, can easily be estimated by using the *M/M/1* property of each node:

$$E[K_i(t)] = \frac{\rho_i}{1-\rho_i}. \tag{11.63}$$

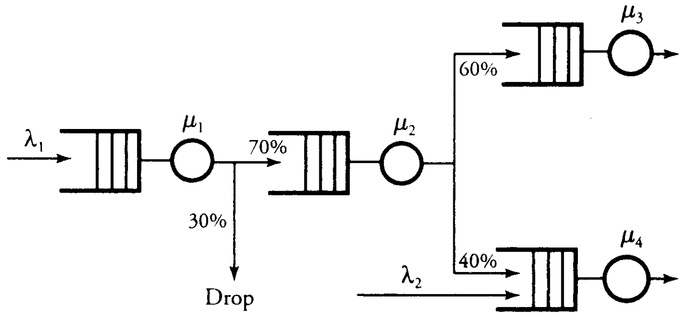


Figure 11.20 A Four-queueing-node system solved using Burke's theorem

Therefore, the expected number of packets in the entire system is derived by

$$E[K(t)] = \sum_{i=1}^{m} E[K_i(t)] = \sum_{i=1}^{m} \frac{\rho_i}{1-\rho_i}. \tag{11.64}$$

We can now use Little's formula to estimate the total queueing delay that a packet should expect in the entire system:

$$E[T] = \frac{1}{\lambda} \sum_{i=1}^{m} E[K_i(t)] = \frac{1}{\lambda} \sum_{i=1}^{m} \frac{\rho_i}{1-\rho_i}. \tag{11.65}$$

This last important result can help estimate the speed of a particular system modeled by a series of queueing nodes.

Example. Figure 11.20 shows four queueing nodes. There are two external sources at rates: λ_1 = 2 million packets per second and λ_2 = 0.5 million packets per second. Assume that all service rates are identical and equal to $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 2.2$ million packets per second. Find the average delay on a packet passing through nodes 1, 2, and 4.

Solution. The figure shows the utilizations as

$$\rho_1 = \frac{\lambda_1}{\mu_1},$$

$$\rho_2 = \frac{0.7\lambda_1}{\mu_2},$$

and

$$\rho_4 = \frac{0.4(0.7)\lambda_1 + \lambda_2}{\mu_4} = \frac{0.28\lambda_1 + \lambda_2}{\mu_4}.$$

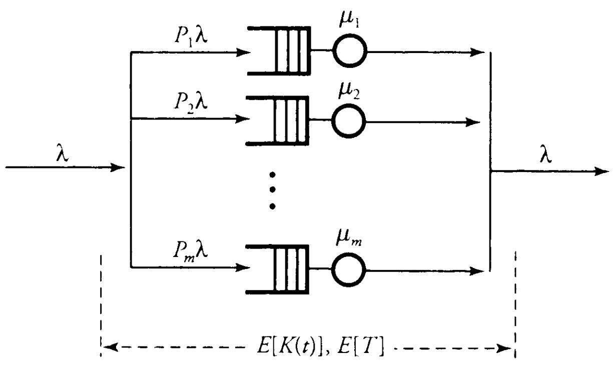


Figure 11.21 Application of Burke's theorem on parallel nodes

The expected numbers of packets in nodes 1, 2 and 4 are, respectively:

$$E[K_1(t)] = \frac{\rho_1}{1-\rho_1} = \frac{\lambda_1}{\mu_1 - \lambda_1},$$

$$E[K_2(t)] = \frac{\rho_2}{1-\rho_2} = \frac{0.7\lambda_1}{\mu_2 - 0.7\lambda_1},$$

and

$$E[K_4(t)] = \frac{\rho_4}{1-\rho_4} = \frac{0.28\lambda_1 + \lambda_2}{\mu_4 - 0.28\lambda_1 - \lambda_2}.$$

The expected overall delay, then, is

$$E[T] = \frac{E[K_1(t)]}{\lambda_1} + \frac{E[K_2(t)]}{0.7\lambda_1} + \frac{E[K_4(t)]}{0.28\lambda_1 + \lambda_2}.$$

Burke's Theorem on Parallel Nodes

We can now easily apply Burke's theorem in cases of in-parallel queueing nodes, as shown in Figure 11.21. Suppose that the incoming traffic with rate λ is distributed over m parallel queueing nodes with probabilities P_1 through P_m, respectively. Also assume that the queuing nodes' service rates are μ_1 through μ_m, respectively. In this system, the utilization for any node i is expressed by

$$\rho_i = \frac{P_i\lambda}{\mu_i}. \tag{11.66}$$

Assuming that each node is represented by an *M/M/1* queueing model, we calculate the expected number of packets in the queue of a given node *i* by

$$E[K_i(t)] = \frac{\rho_i}{1-\rho_i}. \tag{11.67}$$

But the main difference between a cascaded queueing system and a parallel queueing system starts here, at the analysis on average delay. In the parallel system, the overall average number of packets, $\sum_{i=1}^{m} E[K_i(t)]$, is an irrelevant metric. The new argument is that a packet has a random chance of P_i to choose only one of the m nodes (node i) and thus is not exposed to all m nodes. Consequently, we must calculate the average number of queued packets that a new arriving packet faces in a branch i. This average turns out to have a perfect stochastic pattern and is obtained by $E[K(t)]$:

$$E[K(t)] = E[E[K_i(t)]] = \sum_{i=1}^{m} P_i E[K_i(t)] = \sum_{i=1}^{m} P_i\left(\frac{\rho_i}{1-\rho_i}\right). \tag{11.68}$$

Using Little's formula, we can first estimate the delay incurred in branch i:

$$E[T_i] = \frac{E[K_i(t)]}{P_i\lambda}. \tag{11.69}$$

Now, the total queueing delay that a packet should expect in the entire system, $E[T]$, is the average delay over all m parallel branches. This delay is, in fact, a stochastic mean delay of all parallel branches, as follows:

$$\begin{aligned} E[T] &= E[E[T_i]] = \sum_{i=1}^{m} P_i E[T_i] = \sum_{i=1}^{m} P_i \frac{E[K_i(t)]}{P_i\lambda} \\ &= \frac{1}{\lambda}\sum_{i=1}^{m} \frac{\rho_i}{1-\rho_i}. \end{aligned} \tag{11.70}$$

Obviously, $E[T]$ can help us estimate the speed of a particular m parallel-node system. An important note here is that Equations (11.64) and (11.68) may look the same. But the fact is that since the utilization in the case of cascaded nodes, $\rho_i = \lambda_i / \mu_i$, and the one for parallel nodes, $\rho_i = P_i\lambda_i$, are different, the results of $E[K(t)]$ in these cases too are different. We can argue identically on $E[T]$ for the two cases. Although Equations (11.65) and (11.70) are similar, we should expect the overall delay in the parallel-node case to be far less than the one for in-series case, owing to the same reason.

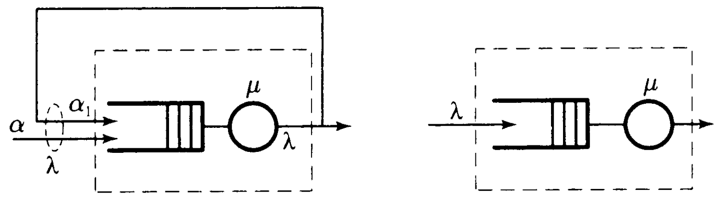


Figure 11.22 A feedback system and its simplified equivalent

11.6.2 Jackson's Theorem

One of the most frequently applied theorems in analyzing a network of queues is known as *Jackson's theorem*, which has several segments. Here, we emphasize the most applicable phase, known as open networks of queues. The essence of Jackson's theorem is applied to situations in which a packet visits a particular queue more than once. In such conditions, the network typically contains *loops*, or *feedback*.

Modeling Feedback in Networks

Figure 11.22 shows a basic cascaded queueing network, including a simple $M/M/1$ queueing element with a server in which simple feedback is implemented. Packets arrive at the left at rate α, passing through a queue and server with service rate μ and are partially fed back to the system at rate α_1. Thus, this queueing architecture allows packets to visit the queue more than once by circulating them back to the system. This system is a model of many practical networking systems. One good example is switching systems that multicast packets step by step, using a recycling technique discussed in later chapters. In Figure 11.22, if the probability that a packet is fed back to the input is p, it is obvious that the total arrival rate to the queue, or λ, is

$$\lambda = \alpha + p\lambda. \tag{11.71}$$

Equation (11.71) demonstrates that the equivalent of the system surrounded by the dashed line can be configured as shown in Figure 11.22. Since $\alpha_1 = p\lambda$, we can combine this equation and Equation 11.71 and derive a relationship between λ and α as

$$\lambda = \frac{\alpha}{1-p}. \tag{11.72}$$

The utilization of the simplified system denoted by $\rho = \frac{\lambda}{\mu}$ is, clearly:

$$\rho = \frac{\alpha}{\mu(1-p)}. \tag{11.73}$$

By having the utilization of the system, we can derive the probability that k packets are in the system, $P[K(t) = k]$, knowing that the system follows the $M/M/1$ rule:

$$P[K(t) = k] = (1-\rho)\rho^k = \frac{\mu(1-p)-\alpha}{\mu(1-p)}\left(\frac{\alpha}{\mu(1-p)}\right)^k. \quad (11.74)$$

The expected number of packets in the system, $E[K(t)]$, can be derived by using the $M/M/1$ property:

$$E[K(t)] = \frac{\rho}{1-\rho} = \frac{\alpha}{\mu(1-p)-\alpha}. \quad (11.75)$$

Using Little's formula, we can now estimate the total queueing and service delay that a packet should expect. Let $E[T_u]$ be the expected delay for the equivalent unit shown in Figure 11.22 with lump sum arrival rate λ. Let $E[T_f]$ be the expected delay for the system with arrival rate α:

$$E[T_u] = \frac{E[K(t)]}{\lambda} = \frac{\alpha}{\lambda\,(\mu(1-p)-\alpha)} \quad (11.76)$$

and

$$E[T_f] = \frac{E[K(t)]}{\alpha} = \frac{1}{\mu(1-p)-\alpha}. \quad (11.77)$$

Note that we should always see the inequity $E[T_u] < E[T_f]$ as expected.

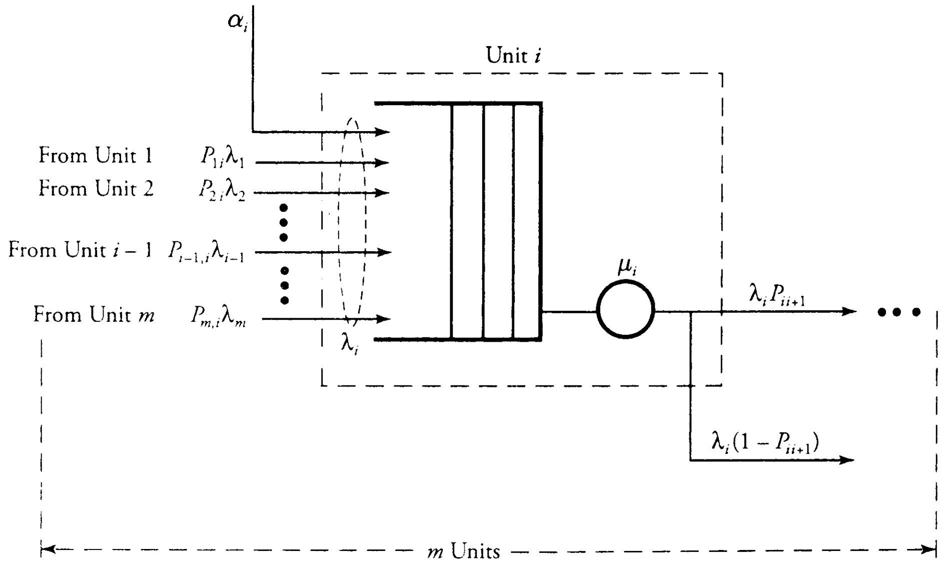
Open Networks

Figure 11.23 shows a generic model of Jackson's theorem when the system is open. In this figure, unit i of an m-unit system is modeled with an arrival rate sum of λ_i and a service rate μ_i. Unit i may receive an independent external traffic of α_i. The unit can receive feedback and feed-forward traffic from $i-1$ other units. At the output of this unit, the traffic may proceed forward with probability P_{ii+1} or may leave the system with probability $1-P_{ii+1}$. The total arrival rate, λ_i, is then

$$\lambda_i = \alpha_i + \sum_{j=1}^{m} P_{ji}\lambda_j. \quad (11.78)$$

The instantaneous total number of packets in all m nodes, K(t), is a vector consist of the number of packets in every individual node and is expressed by

$$\mathrm{K(t)} = \{K_1(t), K_2(t), \cdots, K_m(t)\}. \quad (11.79)$$



α_i
Unit i
From Unit 1 $P_{1i}\lambda_1$
From Unit 2 $P_{2i}\lambda_2$
From Unit $i-1$ $P_{i-1,i}\lambda_{i-1}$
From Unit m $P_{m,i}\lambda_m$
λ_i
μ_i
$\lambda_i P_{ii+1}$
$\lambda_i(1-P_{ii+1})$
m Units

Figure 11.23 Generalization of a node model for Jackson's theorem

The probability that k packets are in the system is the joint probabilities of numbers of packets in m units:

$$P[K(t)=k]=\prod_{i=1}^{m} P[K_i(t)=k_i]. \tag{11.80}$$

The expected number of packets in all units is obtained by

$$E[K(t)]=E[K_1(t)]+E[K_2(t)]+\cdots+E[K_m(t)]. \tag{11.81}$$

Finally, the most significant factor used in evaluating a system of queueing nodes, total delay at stage i, is calculated by applying Little's formula:

$$E[T_i]=\frac{\sum_{i=1}^{m} E[K_i(t)]}{\alpha_i}. \tag{11.82}$$

We assume that α_i is the only input to the entire system. In practice, there may be other inputs to the system, in which case the total delay can be obtained by using *superposition*. Equations (11.80) through (11.82) are the essence of Jackson's theorem on open queueing networks.

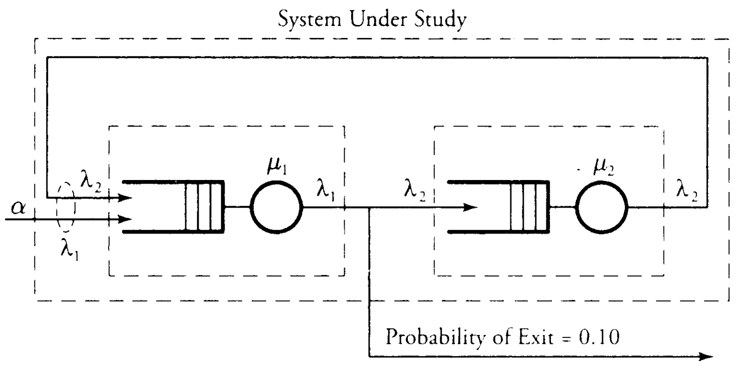


Figure 11.24 Two communication nodes with feedback

Example. Figure 11.24 shows a cascade of two $M/M/1$ nodes in which packets arrive at rate $\alpha = 10,000$ packets per second. Service rates of both nodes are $\mu_1 = \mu_2 = 200,000$ packets per second. At the output of node 1, packets exit with probability of $p = 0.10$; otherwise, they are driven into node 2 and then fed back to node 1. Find the estimated delay incurred on a packet, including the delay from circulations.

Solution. We first calculate the arrival rates to two nodes, λ_1 and λ_2:

$$\lambda_1 = \lambda_2 + \alpha$$

and

$$\lambda_2 = (1 - p)\lambda_1.$$

By applying the values of $\alpha = 10,000$ and $p = 0.10$, this set of equations results in $\lambda_1 = 100,000$ and $\lambda_2 = 90,000$. The utilization for each system can now be derived as $\rho_1 = \lambda_1/\mu_1 = 0.5$ and $\rho_2 = \lambda_2/\mu_2 = 0.45$. According to $M/M/1$ discipline results obtained in the previous chapter, the average number of packets in each system is

$$E[K_1(t)] = \frac{\rho_1}{1 - \rho_1} = 1$$

and

$$E[K_2(t)] = \frac{\rho_2}{1 - \rho_2} = 0.82.$$

The delay is

$$E[T_{u1}] = \frac{E[K_1(t)]}{\lambda_1} = 0.01 \text{ msec},$$

$$E[T_{u2}] = \frac{E[K_2(t)]}{\lambda_2} = 0.009 \text{ msec},$$

and

$$E[T_f] = \frac{E[K_1(t)] + E[K_2(t)]}{\alpha} = 0.18 \text{ msec}.$$

11.7 Summary

This chapter presented the mathematical foundation for basic analysis of computer networks developing packet queues, using Markovian and non-Markovian cases. Single queues are the simplest models for the analysis of a single node in a communication network. *Little's theorem* relates the average number of packets and the average time per packet in the system to the arrival rate.

Birth-and-death processes can help identify the activity within a packet queue. Queueing node models with several scenarios were presented: finite versus infinite queueing capacity, one server versus several servers, and Markovian versus non-Markovian systems. The non-Markovian models are further classified into systems with non-Markovian services and systems with non-Markovian traffic arrivals. On a non-Markovian service model, we assume independent interarrivals and service times, which leads to the $M/G/1$ system.

Burke's theorem is the fundamental tool analyzing networks of packet queues. By Burke's theorem, we can present solutions to a network of several queues. We applied Burke's theorem in-series queueing nodes and in-parallel queueing nodes. *Jackson's theorem* provides a practical solution to situations in which a packet visits a particular queue more than once through *loops* or *feedback*. Open Jackson networks have a product-form solution for the joint queue length distribution in the steady state. We can derive several performance measures of interest, because nodes can be treated independently. Jackson networks with no external arrivals or departures forming a closed network also have a product-form solution for the steady-state joint queue length distribution.

The next chapter presents quality-of-service (QoS) in networks, another important issue of networking. Recall that QoS is a significant parameter in routing packets requiring performance.

11.8 Exercises

1. Each buffered output line of a data demultiplexer receives a block of packets every 20 μs and keeps them in the buffer. A sequencer checks each block for any packet misordering and corrects it, if necessary. It takes 10 μs to identify any packet misordering at each block and 30 μs to make the right sequence if there is any packet misordering. Suppose that a buffer is initially empty and that the numbers of misorderings in the first 15 blocks are 2, 4, 0, 0, 1, 4, 3, 5, 2, 4, 0, 2, 5, 2, 1.
   (a) Illustrate a plot of queueing behavior for the number of packet blocks in the demultiplexer's output line as a function of time.
   (b) Find the mean number of packet blocks in each output line's queue.
   (c) Determine the percentage of the time that a buffer is not empty.

2. A buffered router presented by an $M/M/1$ model receives Markovian traffic with a mean packet-arrival rate of λ=40/sec and an overall utilization of $\rho = 0.9$.
   (a) Calculate the average number of packets in the queueing system.
   (b) Determine the average time a packet remains in the queueing system.
   (c) Sketch the Markov chain of the node's process.

3. For an $M/M/1$ system with service rate μ, the distribution of service time is exponential and is equal to $e^{-\mu t}$. In other words, by letting T be a random variable to represent the time until the next packet departure, $P[T > t] = e^{-\mu t}$. Now, consider an $M/M/a$ system with i servers busy at a time, equal service rate μ per server, and a similar definition of random variable T.
   (a) Show T in terms of $T_1, T_2, \ldots$, where T_i is a random variable representing the time until the next packet departure in server i.
   (b) Find the distribution of service time, $P[T > t]$.

4. Consider a network node represented by an $M/M/1$ model.
   (a) Find the probability that the number of packets in the node is less than a given number k, $P[K(t) < k]$. This is normally used to estimate the threshold time of the network node to reach a certain load.
   (b) If we require that $P[K(t) < 20] = 0.9904$, determine the switching service rate if the arrival rate to this node is 300 packets per second.

5. For an $M/M/1$ queueing system:
   (a) Find $P[K(t) \geq k]$, where k is a constant number.
   (b) Determine the maximum allowable arrival rate in a system with service rate μ, if $P[K(t) \geq 60] = 0.01$ is required.

6. Consider a network node modeled by an $M/M/1$ queue in which a packet's willingness to join the queue is impacted by the queue size. A packet that finds $i \in \{0, 1, \cdots\}$ other packets in the system joins the queue with probability $\frac{1}{i+1}$ and otherwise leaves immediately. If the arrival rate is λ and the mean service rate is μ:
   (a) Sketch the state-transition diagram for this system .
   (b) Develop and solve the balance equations for the equilibrium probabilities p_i.
   (c) Show that a steady-state distribution exists.
   (d) Find the utilization of the server.
   (e) Find the throughput, the average number of packets in the system.
   (f) Find the average response time for a packet that decides to join.

7. In an $M/M/2$ communication node, packets arrive according to a Poisson process of rate 18 per second. The system has two parallel crossbar switches, and each switch spends an exponentially distributed amount of time with mean 100 ms to process a packet.
   (a) Find the probability that an arriving packet must wait to be processed.
   (b) Find the mean number of packets in the system
   (c) Find the mean time a packet spends in the system.
   (d) Find the probability that more than 50 packets are in the system.

8. Consider a high-speed node of a data network having parallel-plane switching fabrics, in which six packets can be processed at a time without any prior waiting time ($M/M/6/6$). Assume that the arrival rate is 100 packets/sec and that the mean service rate is 20 packets/sec.
   (a) What is the probability of blocking a packet?
   (b) How many more switches are required to reduce the blocking probability to 50 percent more?

9. When a wireless user moves to a new cell, a handoff request for a new channel is needed. A handoff can be modeled with traffic type i assumption. When all the channels are used or none are available, the handoff call has to be terminated or blocked in the new base station. Classify the traffic into k types. Each call uses only one channel, and each channel uses only one radio channel. The handoff process at the new base station is modeled using an $M/M/c/c$ system: random interarrival call time, exponential holding time of a channel, c channels, and c handoff calls. Let λ_i be the handoff request rate for traffic type $i \in \{0, 1, ..., k\}$, and let $1/\mu_i$ be the mean channel-exchange time for traffic type i. When j channels are busy,

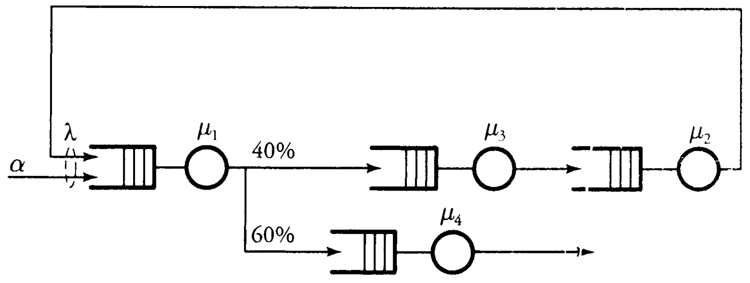


Figure 11.25 Exercise 11 network example

handoff calls depart at rate $j\mu_i$. When all c_i channels are in use, the channel-exchange rate is $c_i\mu_i$. In this case, any new arriving handoff calls are blocked.

(a) Show a continuous-time Markov chain for the model.
(b) Derive a set of global balance equations.
(c) If P_0 is the probability that no channel exchange is requested for traffic type i, find P_j, the probability that j channel exchanges are requested for traffic type i.
(d) Find the handoff blocking probability, P_{c_i}.

10. Continuing the previous exercise, we want to obtain some statistical performance-evaluation results for the handoff.

(a) Assume that the total available numbers of channels (c_i) are 10, 50, and 100, respectively; plot the handoff blocking probability with a choice of three mean holding times of 10 ms, 20 ms, and 30 ms.
(b) Discuss the results obtained in the plots.

11. Consider a router containing four networked queueing nodes, each represented by an $M/M/1$ model, as shown in Figure 11.25. Any percentage number shown on each branch expresses the share of the branch from the traffic coming to its branching point. The arrival rate to the system is $\alpha = 20$ packets per ms. The service rates in these nodes are $\mu_1 = 100$, $\mu_2 = 100$, $\mu_3 = 20$, and $\mu_4 = 30$ packets per ms, respectively.

(a) Find the arrival rate to each of the four queueing units.
(b) Find the average number of packets in each unit.
(c) Find the average delay for a packet from its arrival to the system until it exits the system.

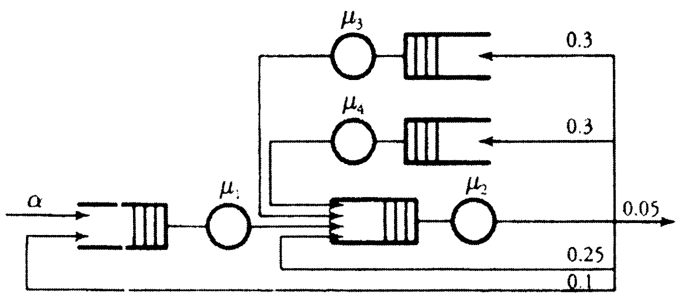


Figure 11.26 Exercise 12 network example

12. Figure 11.26 shows a network of four data switching nodes modeled by four *M/M/1* systems. The arrival rate to this network is λ. The outgoing packets get distributed over the outgoing link and feedback paths with the probabilities indicated on the figure. The arrival rate to the system is $\alpha = 200$ packets per second, and the service rates are $\mu_1 = 4$, $\mu_2 = 3$, $\mu_3 = 5$, and $\mu_2 = 2$ packets per ms.
   (a) Find the arrival rate to each of the four queueing units.
   (b) Find the average number of packets in each unit.
   (c) Find the average system delay for a packet.

13. A network of four switching nodes is modeled by four *M/M/1* systems, as shown in Figure 11.27. The arrival rate to the network is $\alpha = 100$ packets per second. The outgoing packets get distributed over the outgoing link and feedback paths with probabilities given on the figure.
   (a) Find the arrival rate to each of the four queueing units.
   (b) Find the average number of packets in each unit.
   (c) Find the average system delay for a packet.

14. A network of six switching nodes is modeled by six *M/M/1* systems, as shown in Figure 11.28. Each of the five parallel nodes receives a fair share of traffic. The arrival rate to the system is $\alpha = 200$ packets per second, the service rate for the single node is $\mu_0 = 100$ packets per ms, and the service rate for each of the five nodes is $\mu_i = 10$ packets per ms.
   (a) Find the arrival rate to each of the six queueing units.
   (b) Find the average number of packets in each unit.
   (c) Find the average system delay for a packet.

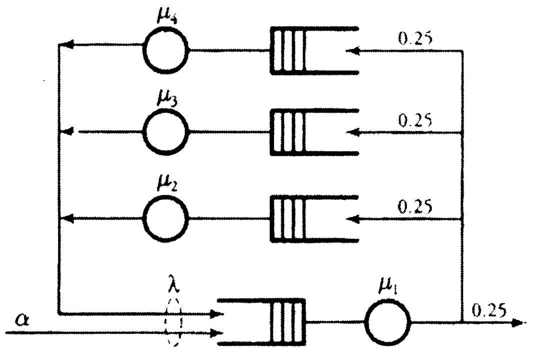


Figure 11.27 Exercise 13 network example

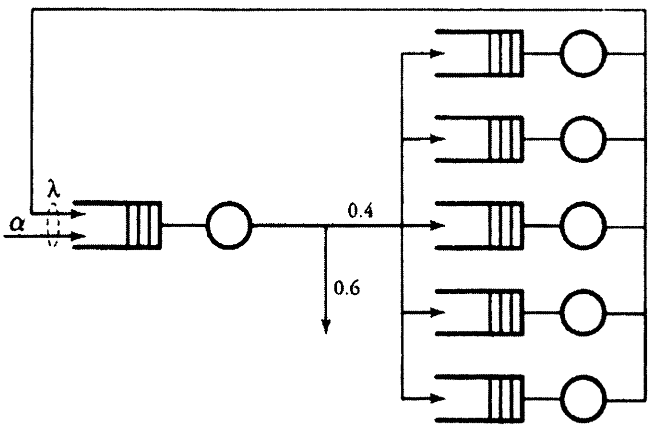


Figure 11.28 Exercise 14 network example

15. A network of four routers is modeled by four *M/M/1* systems, as shown in Figure 11.29. Outgoing packets get distributed over the outgoing link and feedback paths with the probabilities indicated on the figure. The arrival rate to the system is $\alpha = 800$ packets per second, and the service rates are $\mu_1 = 10$, $\mu_2 = 12$, $\mu_3 = 14$, and $\mu_1 = 16$ packets per ms.

(a) Find the arrival rate to each of the four queueing units.

(b) Find the average number of packets in each unit.

(c) Find the average system delay for a packet.

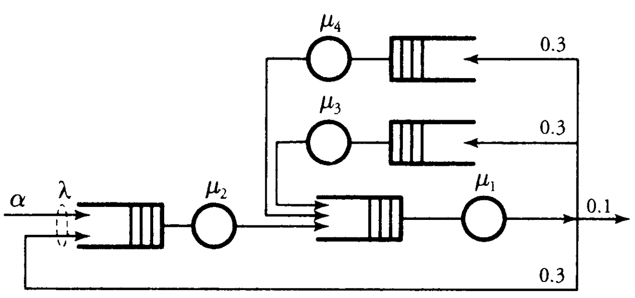


Figure 11.29 Exercise 16 network example

16. *Computer simulation project.* Write a computer program to simulate an input buffer of a router. Consider $K = 64$ buffer slots; each slot can fit only in a packet of size 1,000 bytes.
    (a) Construct and simulate 1 bit of memory.
    (b) Extend your program to construct 1,000 bytes of memory.
    (c) Further extend your program to simulate all 64 buffer slots.
    (d) Dynamically assign packets of different sizes every 1 ms, and send out the packet every t seconds. Measure the performance metrics of this buffer given different values of t

17. *Computer simulation project.* Carry the program you developed for a single buffer in the previous project and extend it to simulate two connected buffers: one with $K_1 = 64$ buffer slots and the other one with $K_2 = 128$ buffer slots. Each slot can fit only in a packet of size 1,000 bytes. Dynamically assign packets of different size every 1 ms, and send out the packet every t_1 seconds from the first buffer and every t_2 seconds from the second buffer:
    (a) Construct and simulate each of the two buffers individually.
    (b) Integrate these two buffers and measure the performance metrics of both buffers together, given different values of t_1 and t_2.

CHAPTER 12

Quality of Service and Resource Allocation

Recall from Chapter 3 that the main portion of the port processor interface in routers supports *quality of service* (QoS). This chapter explores all angles of QoS and the numerous methods of providing it at various protocol stack layers. This chapter covers the following topics:

- *Overview of QoS*
- *Integrated services QoS*
- *Traffic shaping*
- *Admission control*
- *Reservation protocol (RSVP)*
- *Differentiated services QoS*
- *Resource allocation*

Two broad approaches to QoS are ***integrated services*** and ***differentiated services***. Integrated services provide QoS to individual applications and flow records. Providing QoS requires certain features to be maintained in switching nodes. QoS protocols govern traffic shaping and packet scheduling. Traffic shaping regulates the spacing between incoming packets. Other advanced QoS protocols covered are admission control and RSVP.

The differentiated services provide QoS support to a broad class of applications. The basics of resource allocation for packet-switched networks are reviewed, as is resource allocation for all possible layers of the protocol stack. Resource-allocation algorithms can be used to avoid possible ATM congestion.

12.1 Overview of QoS

Communication networks face a variety of quality-of-service demands. The main motivation for a QoS unit in a data network port processor is to control access to available bandwidth and to regulate traffic. Traffic regulation is always necessary in WANs in order to avoid congestion. A network must be designed to support both real-time and non-real-time applications. Voice and video transmissions over IP must be able to request a higher degree of assurance from the network. A network that can provide these various levels of services requires a more complex structure.

The provision of QoS to a network either does or does not come with a guarantee. Nonguaranteed QoS is typically based on the *best-effort model*, whereby a network provides no guarantees on the delivery of packets but makes its best effort to do so. In a non-real-time application, a network can use the retransmit strategy for successful data delivery. However, in a real-time application, such as voice or video networking, where timely delivery of packets is required, the application requires a low-latency communication. Consequently, the network must be able to handle packets of such applications more carefully.

Approaches to providing quality support can be further divided into *integrated services* or *differentiated services*. The details of these two approaches are discussed in Sections 12.2 and 12.3, respectively.

12.2 Integrated Services QoS

The *integrated services approach*, consisting of two service classes, defines both the service class and mechanisms that need to be used in routers to provide the services associated with the class. The first service class, the *guaranteed service class*, is defined for applications that cannot tolerate a delay beyond a particular value. This type of service class can be used for real-time applications, such as voice or video communications. The second, *controlled-load service class*, is used for applications that can tolerate some delay and loss. Controlled-load service is designed such that applications run very well when the network is not heavily loaded or congested. These two service classes cover the wide range of applications on the Internet.

A network needs to obtain as much information as possible about the flow of traffic to provide optimum services to the flow. This is true especially when real-time services to

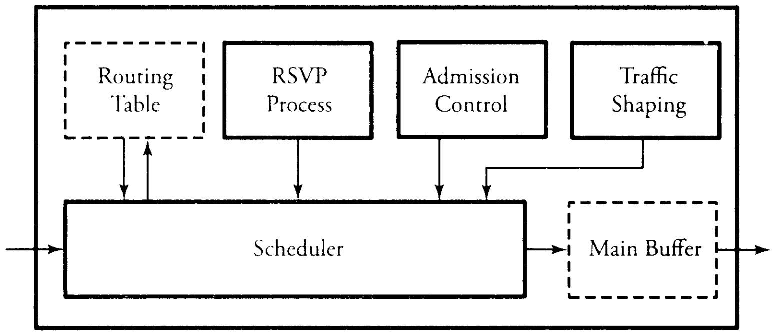


Figure 12.1 Overview of QoS methods in integrated services

an application are requested. Any application request to a network has to first specify the type of service required, such as controlled load or guaranteed service. An application that requires guaranteed service also needs to specify the maximum delay that it can tolerate. Once the delay factor is known to a service provider, the QoS unit of the node must determine the necessary processes to be applied on incoming flows. Figure 12.1 shows four common categories of processes providing quality of service.

1. *Traffic shaping* regulates turbulent traffic.
2. *Admission control* governs whether the network, given information about an application's flow, can admit or reject the flow.
3. *Resource allocation* lets network users reserve bandwidth on neighboring routers.
4. *Packet scheduling* sets the timetable for the transmission of packet flows. Any involving router needs to queue and transmit packets for each flow appropriately.

The widespread deployment of the integrated services approach has been deterred owning to scalability issues. As the network size increases, routers need to handle larger routing tables and switch larger numbers of bits per second. In such situations, routers need to refresh information periodically. Routers also have to be able to make admission-control decisions and queue each incoming flow. This action leads to scalability concerns, especially when networks scale larger.

12.2.1 Traffic Shaping

Realistically, spacing between incoming packets has an irregular pattern, which in many cases causes congestion. The goal of *traffic shaping* in a communication network is to control access to available bandwidth to regulate incoming data to avoid congestion,

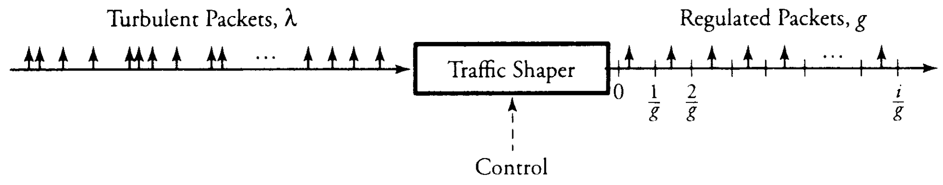


Figure 12.2 Traffic shaping to regulate any incoming turbulent traffic

and to control the delay incurred by packets (see Figure 12.2). Turbulent packets at rate λ and with irregular arrival patterns are regulated in a traffic shaper over equal-sized $1/g$ intervals.

If a policy dictates that the packet rate cannot exceed a specified rate even though the network node's access rates might be higher, a mechanism is needed to smooth out the rate of traffic flow. If different traffic rates are applied to a network node, the traffic flow needs to be regulated. (Monitoring the traffic flow is called *traffic policing*.) Traffic shaping also prevents packet loss by preventing the sudden increased usage of system bandwidth. The stochastic model of a traffic shaper consists of a system that converts any form of traffic to a deterministic one. Two of the most popular traffic-shaping algorithms are *leaky bucket* and *token bucket*.

Leaky-Bucket Traffic Shaping

This algorithm converts any turbulent incoming traffic into a smooth, regular stream of packets. Figure 12.3 shows how this algorithm works. A leaky-bucket interface is connected between a packet transmitter and the network. No matter at what rate packets enter the traffic shaper, the outflow is regulated at a constant rate, much like the flow of water from a leaky bucket. The implementation of a leaky-bucket algorithm is not difficult.

At the heart of this scheme is a finite queue. When a packet arrives, the interface decides whether that packet should be queued or discarded, depending on the capacity of the buffer. The number of packets that leave the interface depends on the protocol. The packet-departure rate expresses the specified behavior of traffic and makes the incoming bursts conform to this behavior. Incoming packets are discarded once the bucket becomes full.

This method directly restricts the maximum size of a burst coming into the system. Packets are transmitted as either fixed-size packets or variable-size packets. In the fixed-size packet environment, a packet is transmitted at each clock tick. In the variable-size packet environment, a fixed-sized block of a packet is transmitted. Thus, this algorithm

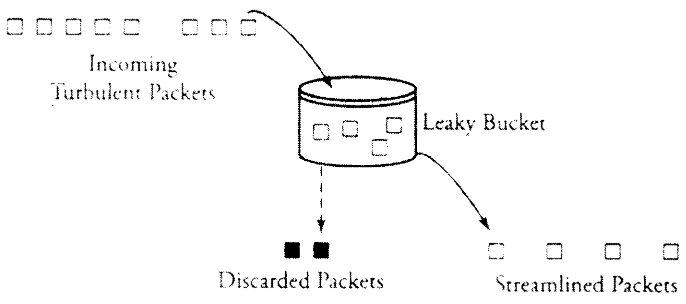


Figure 12.3 The leaky-bucket traffic-shaping algorithm

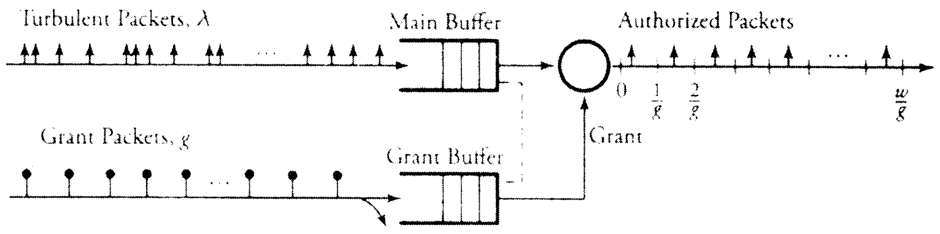


Figure 12.4 Queueing model of leaky-bucket traffic-shaping algorithm

is used for networks with variable-length packets and also equal-sized packet protocols, such as ATM.

The leaky-bucket scheme is modeled by two main buffers, as shown in Figure 12.4. One buffer forms a queue of incoming packets, and the other one receives authorizations. The leaky-bucket traffic-shaper algorithm is summarized as follows.

Begin Leaky-Bucket Algorithm

1. **Define for the Algorithm:**
   λ = rate at which packets with irregular rate arrive at the *main buffer*
   g = rate at which *authorization grants* arrive at the *grant buffer*
   w = size of the grant buffer and can be dynamically adjusted
2. Every $1/g$ seconds, a grant arrives.
3. Over each period of i/g seconds, i grants can be assigned to the first i incoming packets, where $i \leq w$, and packets exit from the queue one at a time every $1/g$ seconds, totaling i/g seconds.

4. If more than w packets are in the main buffer, only the first w packets are assigned grants at each window time of $1/g$, and the rest remain in the main queue to be examined in the next $1/g$ interval.
5. If no grant is in the grant buffer, packets start to be queued.

With this model, w is the bucket size. The bucket in this case is the size of the window that the grant buffer opens to allow w packets from the main buffer to pass. This window size can be adjusted, depending on the rate of traffic. If w is too small, the highly turbulent traffic is delayed by waiting for grants to become available. If w is too large, long turbulent streams of packets are allowed into the network. Consequently, with the leaky-bucket scheme, it would be best for a node to dynamically change the window size (bucket size) as needed. The dynamic change of grant rates in high-speed networks may, however, cause additional delay through the required feedback mechanism.

Figure 12.5 shows the Markov chain state diagram (see Section C.5) depicting the activity of grant generation in time. At the main buffer, the mean time between two consecutive packets is $1/\lambda$. At the grant buffer, a grant arrives at rate g. Hence, every $1/g$ seconds, a grant is issued to the main buffer on times 0, $1/g$, $2/g$, ... If the grant buffer contains w grants, it discards any new arriving grants. A state $i \in \{0, 1, \ldots w\}$ of the Markov chain refers to the situation in which i grants are allocated to i packets in the main buffer, and thus $w - i$ grants are left available. In this case, the i packets with allocated grants are released one at a time every $1/g$ seconds. When the packet flow is slow, the grant queue reaches its full capacity, and thus grants $(w + 1)$ and beyond are discarded.

We define the following main variables for delay analysis:

- $P_X(x)$ = The probability of x packet arrivals in $1/g$ seconds
- P_i = The probability that i grants have already been assigned to i packets in the main buffer or that the Markov chain is in state i
- P_{ji} = The transition probability from any state j to a state i on the Markov chain

As shown in Figure 12.5, the chain starts at state 0, implying that w grants are available in the grant buffer and that no packet is in the main buffer. In this state, the first arriving packet to the main buffer with probability $P_X(1)$ gets a grant while a new grant arrives in the same period. This creates no changes in the state of the chain. Therefore, the transition probability at state 0 has two components, as follows:

$$P_{00} = P_X(0) + P_X(1). \tag{12.1}$$

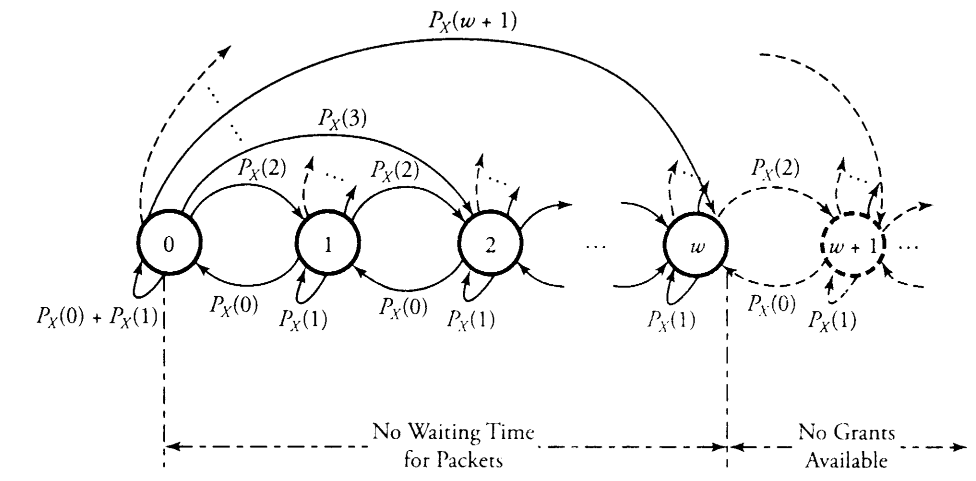


Figure 12.5 State diagram modeling the leaky-bucket traffic-shaping algorithm

This means that P_{00} is equal to the probability that no packet or only one packet arrives $(P_X(0) + P_X(1))$. As long as $i \geq 1$, state 0 can be connected to any state $i \leq w$, inducing the property of the queueing system in Figure 12.5 by

$$P_{0i} = P_X(i+1) \quad \text{for } i \geq 1. \tag{12.2}$$

For example, two new packets arriving during a period $1/g$ with probability $P_X(2)$ change state 0 to state 1 so that one of the two packets is allocated a grant, and thus the chain moves to state 1 from state 0, and the second packet is allocated the new arriving grant during that period of time. The remaining transition probabilities are derived from

$$P_{ji} = \begin{cases} P_X(i-j+1) & \text{for } 1 \leq j \leq i+1 \\ 0 & \text{for } j > i+1 \end{cases}. \tag{12.3}$$

Now, the global balance equations can be formed. We are particularly interested in the probability of any state i denoted by P_i. The probability that the chain is in state 0, P_0, implying that no grant has arrived, is the sum of incoming transitions:

$$P_0 = P_X(0)P_1 + \left[P_X(0) + P_X(1)\right] P_0. \tag{12.4}$$

For P_1, we can write

$$P_1 = P_X(2)P_0 + P_X(1)P_1 + P_X(0)P_2. \tag{12.5}$$

For all other states, the probability of the state can be derived from the following generic expression:

$$P_i = \sum_{j=0}^{i+1} P_X(i - j + 1)P_j \quad \text{for } i \geq 1. \tag{12.6}$$

The set of equations generated from Equation (12.6) can be recursively solved. Knowing the probability of each state at this point, we can use Little's formula (see Section 11.1) to estimate the average waiting period to obtain a grant for a packet, $E[T]$, as follows:

$$E[T] = \frac{\sum_{i=w+1}^{\infty}(i - w)P_i}{g}. \tag{12.7}$$

It is also interesting to note that the state of the Markov chain can turn into "queueing of packets" at any time, as shown by dashed lines in Figure 12.5. For example at state 0, if more than $w + 1$ packets arrive during $1/g$, the state of the chain can change to any state after w, depending on the number of arriving packets during $1/g$. If this happens, the system still assigns $w + 1$ grants to the first $w + 1$ packets and assigns no grant to the remaining packets. The remaining packets stay pending to receive grants.

Example. The probability that x packet arrives in $1/g$ seconds is obtained by

$$P_X(x) = \frac{(\frac{\lambda}{g})^x e^{-\frac{\lambda}{g}}}{x!}.$$

Find the probability that the system has not issued any grants.

Solution. We give the first final result, which starts at P_0, and leave the rest of this recursive effort to the reader. The final form of P_0, using Equation (12.4) and $P_X(x)$, starts at:

$$P_0 = \frac{g - \lambda}{g P_X(0)},$$

where this equation can be rearranged to $g = \lambda/(1 - P_0 P_X(0))$. We can then state that the system is considered stable if $\lambda < g$ or $P_0 P_X(0) > 1$.

Token-Bucket Traffic Shaping

Traffic flow for most applications varies, with traffic bursts occurring occasionally. Hence, the bandwidth varies with time for these applications. However, the network must be informed about the bandwidth variation of the flow in order to perform

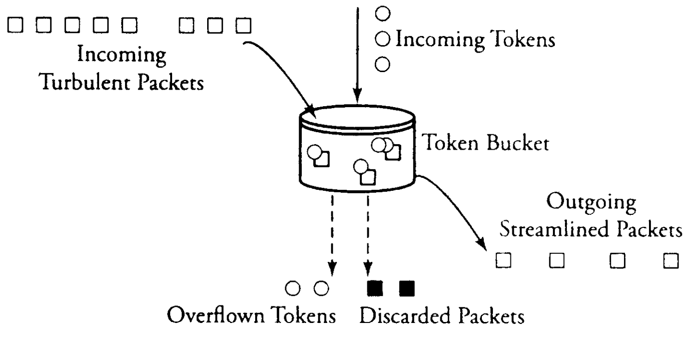


Figure 12.6 The token-bucket traffic-shaping method

admission control and to operate more efficiently. Two parameters—the ***token arrival*** rate, v, and the ***bucket depth***, b—together describe the operation of the token-bucket traffic shaper. The token rate, v, is normally set to the average traffic rate of the source. The bucket depth, b, is a measure of the maximum amount of traffic that a sender can send in a burst.

With the ***token-bucket traffic-shaping*** algorithm, flow traffic is shaped efficiently as shown in Figure 12.6. The token-bucket shaper consists of a buffer, similar to a water bucket, that accepts fixed-size ***tokens*** of data generated by a token generator at a constant rate every clock cycle. Packets with any unpredictable rate must pass through this token-bucket unit. According to this protocol, a sufficient number of tokens are attached to each incoming packet, depending on its size, to enter the network. If the bucket is full of tokens, additional tokens are discarded. If the bucket is empty, incoming packets are delayed (buffered) until a sufficient number of tokens is generated. If the packet size is too big, such that there are not enough tokens to accommodate it, a delay of input packets is carried over.

The best operation of this algorithm occurs when the number of tokens is larger than the number of incoming packet sizes. This is not a big flaw, as tokens are periodically generated. The output rate of packets attached with tokens obeys the clock, and so the corresponding flow becomes a regulated traffic stream. Thus, by changing the clock frequency and varying the token-generation rate, we can adjust the input traffic to an expected rate.

Comparison of Leaky-Bucket and Token-Bucket Approaches

Both algorithms have their pros and cons. The token-bucket algorithm enforces a more flexible output pattern at the average rate, no matter how irregular the incoming traffic is. The leaky-bucket method enforces a more rigid pattern. Consequently, the token bucket is known as a more flexible traffic-shaping method but has greater system complexity. The token-bucket approach has an internal clock that generates tokens with respect to the clock rate. A queueing regulator then attaches incoming packets to tokens as packets arrive, detaches tokens as packets exit, and, finally, discards the tokens. This process adds considerable complexity to the system. In the leaky-bucket approach, no virtual tokens are seen, greatly increasing the speed of operation and enhancing the performance of the communication node. Our primary goal of coming up with a high-speed switching network would thus be easily attained.

12.2.2 Admission Control

The *admission-control* process decides whether to accept traffic flow by looking at two factors:

1. r_s = type of service requested
2. t_s = required bandwidth information about the flow

For controlled-load services, no additional parameters are required. However, for guaranteed services, the maximum amount of delay must also be specified. In any router or host capable of admission control, if currently available resources can provide service to the flow without affecting the service to other, already admitted flows, the flow is admitted. Otherwise, the flow is rejected permanently. Any admission-control scheme must be aided by a policing scheme. Once a flow is admitted, the policing scheme must make sure that the flow confirms to the specified t_s. If not, packets from these flows become obvious candidates for packet drops during a congestion event. A good admission-control scheme is vital to avoid congestion.

12.2.3 Resource Reservation Protocol (RSVP)

The *Resource Reservation Protocol* (RSVP) is typically used to provide real-time services over a connectionless network. RSVP is a soft-state protocol so that it can handle link failure efficiently. The protocol adopts a receiver-oriented approach to resource reservation. This process is required to provide the desired QoS for an application. RSVP supports both unicast and multicast flows. Unlike connection-oriented resource set-up mechanisms, RSVP can adapt to router failures by using an alternative path.

In order for a receiver to make a reservation at intermediate routers, the sender initially sends the t_s message, which passes through each intermediate router before reaching the receiver. This way, the receiver becomes aware of the information on the flow and the path and makes a reservation at each router. This message is sent

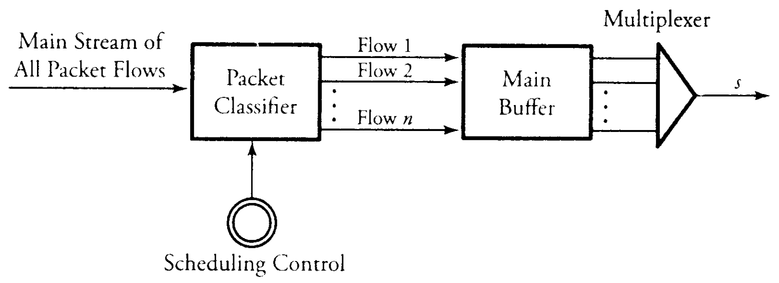


Figure 12.7 Overview of a packet scheduler

periodically to maintain the reservation. The router can accept or deny the reservation, based on its available resources. Also, the t_s message is periodically refreshed to adapt to link failures. When a particular link fails, the receiver receives this message over a different path. The receiver can then use this new path to establish a new reservation; consequently, the network operation continues as usual.

Reservation messages from multiple receivers can be merged if their delay requirements are similar. If a receiver needs to receive messages from multiple senders, it requests a reservation meeting the total of t_s requirements of all senders. Thus, RSVP uses a receiver-oriented approach to request resources to meet the QoS requirements of various applications.

12.2.4 Packet Scheduling

RSVP enables network users to reserve resources at the routers. Once the resources have been reserved, a *packet-scheduling* mechanism has to be in place to provide the requested QoS. Packet scheduling involves managing packets in queues to provide the QoS associated with the packet, as shown in Figure 12.7. A *packet classifier* is the heart of scheduling scheme and performs on the basis of the header information in the packet. Packet classifying involves identifying each packet with its reservation and ensuring that the packet is handled correctly.

Packets are classified according to the following parameters:

- Source/destination IP address
- Source/destination port
- Packet flow priority
- Protocol type
- Delay sensitivity

Based on that information, packets can be classified into those requiring guaranteed services and those requiring controlled-load services. Designing optimal queueing mechanisms for each of these categories is important in providing the requested QoS. For provision of a guaranteed service, *weighted-fair queueing* is normally used. This type of queueing is very effective. For controlled-load services, simpler queueing mechanisms can be used.

Packet scheduling is necessary for handling various types of packets, especially when *real-time packets* and *non-real-time packets* are aggregated. Real-time packets include all delay-sensitive traffic, such as video and audio data, which have a low delay requirement. Non-real-time packets include normal data packets, which do not have a delay requirement. Packet scheduling has a great impact on the QoS guarantees a network can provide. If a router processes packets in the order in which they arrive, an aggressive sender can occupy most of the router's queueing capacity and thus reduce the quality of service. Scheduling ensures fairness to different types of packets and provides QoS. Some of the commonly used scheduling algorithms that can be implemented in the input port processors (IPPs) and output port processors (OPPs) of routers or main hosts are discussed in the next sections.

First-In, First-Out Scheduler

Figure 12.8 shows a simple scheduling scheme: *first in, first out* (FIFO). With this scheme, incoming packets are served in the order in which they arrive. Although FIFO is very simple from the standpoint of management, a pure FIFO scheduling scheme provides no fair treatment to packets. In fact, with FIFO scheduling, a higher-speed user can take up more space in the buffer and consume more than its fair share of bandwidth. FIFO is simple to implement from the hardware standpoint and is still the most commonly implemented scheduling policy, owing to its simplicity. With smart buffer-management schemes, it is possible to control bandwidth sharing among different classes and traffic. Delay bound of this scheduler, T_q, is calculated based on the queueing buffer size, as follows:

$$T_q \leq \frac{K}{s}, \tag{12.8}$$

where K is the maximum buffer size, and s is the outgoing link speed. Certain control algorithms, RSVP, can be used to reserve the appropriate link capacity for a particular traffic flow. However, when high-speed links are active, tight delay bounds are required by real-time applications.

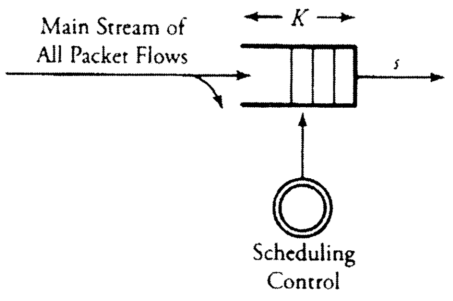


Figure 12.8 A typical FIFO queueing scheduler

Priority Queueing Scheduler

The *priority queueing* (PQ) scheduler combines the simplicity of FIFO schedulers with the ability to provide service classes. With various priority queueing, packets are classified on the priority of service indicated in the packet headers. Figure 12.9 shows a simple model of this scheduler. Lower-priority queues are serviced only after all packets from the higher-priority queues are serviced. Only a queue with the highest priority has a delay bound similar to that with FIFO. Lower-priority queues have a delay bound that includes the delays incurred by higher-priority queues. As a result, queues with lower priorities are subject to bandwidth starvation if the traffic rates for higher-priority queues are not controlled.

In this scheduling scheme, packets in lower-priority queues are services after all the packets from higher-priority queues are transmitted. In other words, those queues with lower priorities are subject to severely limited bandwidth if traffic rates for the higher-priority queues are not controlled.

Priority queueing is either *nonpreemptive* or *preemptive*. In order to analyze these two disciplines, we need to define a few common terms. For a queue with flow i (class i queue), let λ_i, μ_i, and $\rho_i = \lambda_i/\mu_i$ be the arrival rate, mean service rate, and mean offered load (utilization), respectively. We also express a lower value of i for a higher-priority class.

Non-Preemptive Priority Queues. In nonpreemptive priority-queueing schemes, the process of lower-priority packets cannot be interrupted under any condition. Every time an in-service packet terminates from its queue, the waiting packet from the highest-priority queue enters service. This fact makes the system simple from the

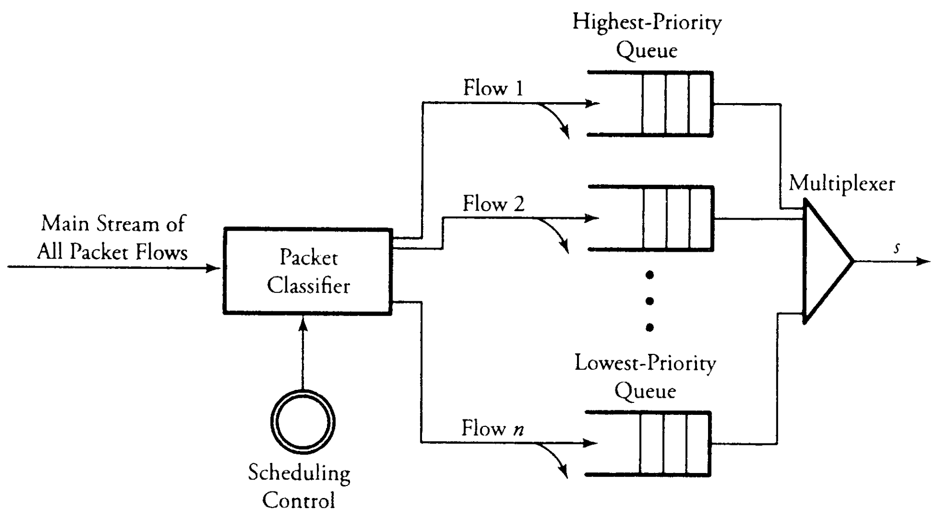


Figure 12.9 A typical priority-queueing scheduler

implementation standpoint. Let $E[T_i]$ be the mean waiting time for a packet in flow i (class i) queue. This total queuing delay has three components, as follows:

1. The mean waiting time for any class i packet until any in-service class j packet among n classes terminates. We denote this time by W_x.
2. The mean time until a packet from a class i or lower (higher-priority) waiting ahead is serviced. We denote this time by $E[T_{q,i}]_2$.
3. The mean time owing to higher-priority packets arriving while the current packet is waiting and being serviced before the current packet. We denote this time by $E[T_{q,i}]_3$.

Hence, $E[T_{q,i}]$ can be the sum of all these three components:

$$E[T_{q,i}] = W_x + E[T_{q,i}]_2 + E[T_{q,i}]_3. \tag{12.9}$$

The first component is computed by using the mean residual service time, r_j, for a class j packet currently in service as:

$$W_x = \sum_{j=1}^{n} \rho_j r_j. \tag{12.10}$$

Using Little's formula, we can derive the second component, $E[T_{q,i}]_2$, by

$$E[T_{q,i}]_2 = \sum_{j=1}^{i} \frac{E[K_{q,j}]}{\mu_j}, \tag{12.11}$$

where $E[K_{q,j}]$ is the mean number of waiting packets and μ_j is the mean service rate for class j packets. Since we can also write $E[T_{q,i}] = E[K_{q,i}]/\mu_i$, we can rewrite Equation (12.11) as

$$E[T_{q,i}]_2 = \sum_{j=1}^{i} \rho_j E[T_{q,j}]. \tag{12.12}$$

Similarly, the third component, $E[T_{q,i}]_3$, can be developed by

$$E[T_{q,i}]_3 = \sum_{j=1}^{i-1} E[T_{q,i}]\left(\frac{\lambda_j}{\mu_j}\right) = E[T_{q,i}]\sum_{j=1}^{i-1} \rho_j. \tag{12.13}$$

By incorporating Equations (12.10), (12.12), and (12.13) into Equation (12.9), we obtain

$$E[T_{q,i}] = \sum_{j=1}^{n} \rho_j r_j + \sum_{j=1}^{i} \rho_j E[T_{q,j}] + E[T_{q,i}]\sum_{j=1}^{i-1} \rho_j. \tag{12.14}$$

Note that the total system delay for a packet passing queue i and the server—in this case, the server is the multiplexer—is

$$E[T_i] = E[T_{q,i}] + \frac{1}{\mu_i}. \tag{12.15}$$

Recall that the $E[T_i]$ expression corresponds to Equation (11.22).

Preemptive Priority Queues. A preemptive-priority queue is a discipline in which service in a lower-priority packet can be interrupted by an incoming higher-priority packet. Let $E[T_i]$ be the mean waiting time for a packet in flow i (class i) queue. Figure 12.10 shows an example in which a class i packet has been interrupted three times by higher-priority packets. This total queueing delay has four components. The first three are identical to those for the nonpreemptive case. The fourth component, θ_i, is the total mean completion time for the current class i packet when it is preempted in the server by higher-priority packets (classes $i - 1$ to 1). Therefore, by including θ_i into

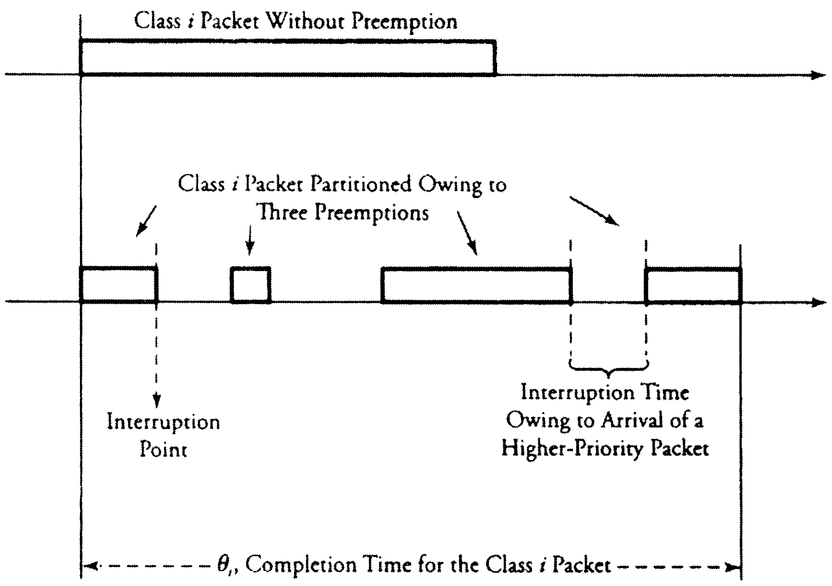

Class i Packet Without Preemption
Class i Packet Partitioned Owing to Three Preemptions
Interruption Point
Interruption Time Owing to Arrival of a Higher-Priority Packet
θ_i, Completion Time for the Class i Packet

Figure 12.10 Preemptive-priority queueing. A class i packet has been interrupted three times by higher-priority packets.

Equation (12.15), we calculate the total system delay for a packet passing queue i and the server as

$$E[T_i] = E[T_{q,i}] + \theta_i. \tag{12.16}$$

During θ_i, as many as $\lambda_j\theta_i$ new higher-priority packets of class j arrive, where $j \in \{1, 2, \ldots i-1\}$. Each of these packets delays the completion of our current class i packet for $1/\mu_j$. Thereore, θ_i, which also includes the service time of the class i packet, as seen in Figure 12.10, can be clearly realized as

$$E[T_i] = E[T_{q,i}] + \theta_i. \tag{12.17}$$

From Equation (12.17), we can compute θ_i as

$$\theta_i = \frac{1}{\mu_i\left(1 - \sum_{j=1}^{i-1}\rho_j\right)}. \tag{12.18}$$

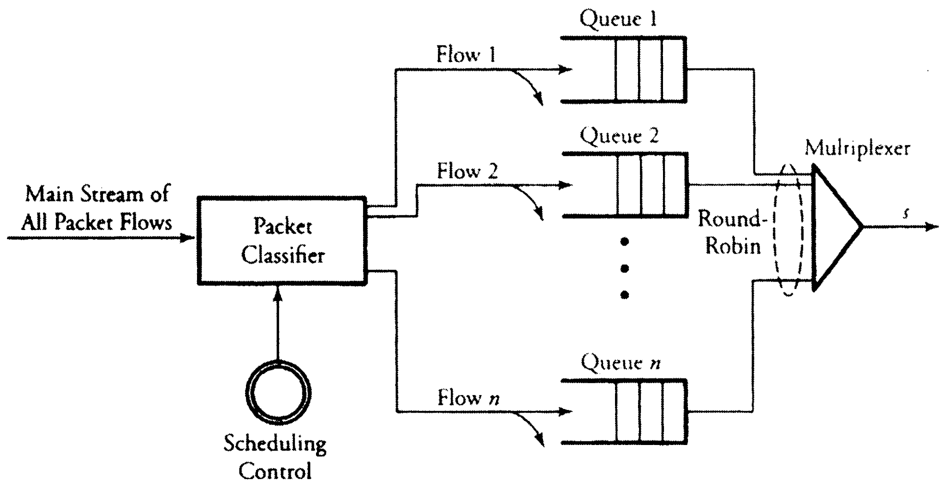


Figure 12.11 Overview of fair-queueing scheduler

If we substitute θ_i of Equation (12.18) into Equation (12.16), we obtain the total system delay for a packet passing queue i and the server:

$$E[T_i] = E[T_{q,i}] + \frac{1}{\mu_i\left(1-\sum_{j=1}^{i-1}\rho_j\right)}, \tag{12.19}$$

where $E[T_{q,\,i}]$ is the same as the one computed for the nonpreemptive case in Equation (12.14).

Fair Queueing Scheduler

The *fair-queueing* (FQ) scheduler is designed to better and more fairly treat servicing packets. Consider multiple queues with different priorities. The fair queueing shown in Figure 12.11 eliminates the process of packet priority sorting. This improves the performance and speed of the scheduler significantly. As a result, each flow i is assigned a separate queue i. Thus, each flow i is guaranteed a minimum fair share of s/n bits per second on the output, where s is the transmission bandwidth, and n is the number of flows (or number of queues). In practice, since all the inputs (flows) may not necessarily have packets at the same time, the individual queue's bandwidth share will be higher than s/n.

Assume that a_j is the arriving time of packet j of a flow. Let s_j and f_j be the start and ending times, respectively, for transmission of the packet. Then, c_j, as the virtual

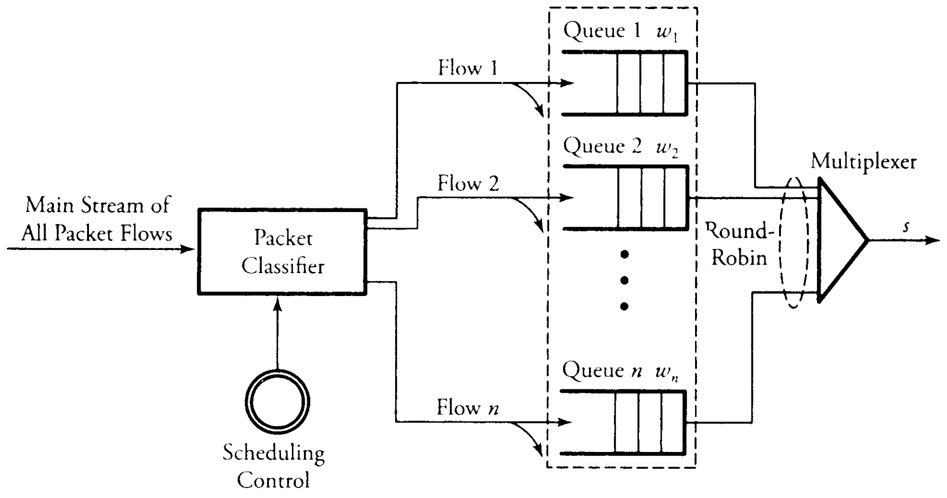


Figure 12.12 Overview of weighted fair-queueing scheduler

clock count needed to transmit the packet, can be obtained from $c_j = f_j - s_j$, since $s_j = \max(f_{i-1}, a_i)$. Then:

$$c_j = f_j - \max(f_{j-1}, a_j). \tag{12.20}$$

We can calculate f_j for every flow and use it as a timestamp. A timestamp is used for maintaining the order of packet transmissions. For example, the timestamp is each time the next packet is to be transmitted. In such a case, any packet with the lowest timestamp finishes the transmission before all others.

Weighted Fair Queueing Scheduler

The *weighted fair-queueing* scheduler (WFQ) scheduler is an improvement over the fair-queueing scheduler. Consider the weighted fair-queueing scheme shown in Figure 12.12. For an n-queue system, queue $i \in \{1 \cdots n\}$ is assigned a weight w_i. The outgoing link capacity s is shared among the flows with respect to their allocated weights. Thus, each flow i is guaranteed to have a service rate of at least

$$r_i = \left(\frac{w_i}{\sum_{j=1}^{n} w_j} \right) s. \tag{12.21}$$

Given a certain available bandwidth, if a queue is empty at a given time, the unused portion of its bandwidth is shared among the other active queues according to their respective weights. WFQ provides a significantly effective solution for servicing real-time packets and non-real-time packets, owing to its fairness and ability to satisfy real-time constraints. Each weight, w_i, specifies the fraction of the total output port bandwidth dedicated to flow *i*. Assigning a higher weight to the real-time queues reduces delay for the real-time packets, so real-time guarantees can be ensured. These weights can be tuned to specific traffic requirements, based on empirical results. As mentioned, any unused bandwidth from inactive queues can be distributed among the active queues with respect to their allocated weights, again with the same fair share of

$$r_i = \left(\frac{w_i}{\sum_{j \in b(t)} w_j} \right) s, \tag{12.22}$$

where *b(t)* is a set of active queues at any time *t*. The delay bound in this scheme is independent of the maximum number of connections *n*, which is why WFQ is considered one of the best queueing schemes for providing tight delay bounds. As a trade-off to the effectiveness of this scheme, the implementation of a complete weighted fair-queueing scheduler scheme is quite complex.

A version of this scheduler, the so-called *weighted round-robin* (WRR) scheduler, was proposed for asynchronous transfer mode. In WRR, each flow is served in a round-robin fashion with respect to a weight assigned for each flow *i*, without considering packet length. This is very similar to the deficit round-robin scheduler discussed next for fixed-size packets. Since the system deals only with same-size packets, it is very efficient and fast. The service received by an active flow during its round of service is proportional to its fair share of the bandwidth specified by the weight for each flow.

Example. Figure 12.13 shows a set of four flows—A, B, C, and D—to be processed at one of the outputs of a router. Each packet's arrival time and label are indicated. Three schedulers are compared: priority queueing, fair queueing, and weighted fair queueing. All schedulers scan the flows, starting from flow A. With priority queueing, priorities decrease from line A to line D. With fair queueing, if the arrival times of two packets are the same, the smaller flow number is selected. Using weighted fair queueing, we assumed that flows A, B, C, and D are given 20 per cent, 10 per cent, 40 per cent, and 30 per cent of the output capacity, respectively.

Deficit Round-Robin Scheduler

In weighted fair queueing the dependency on a sorting operation that grows with the number of flows is a concern for scalability as speed increases. In the *deficit round-robin* (DRR) scheduler, each flow *i* is allocated b_i bits in each round of service, and

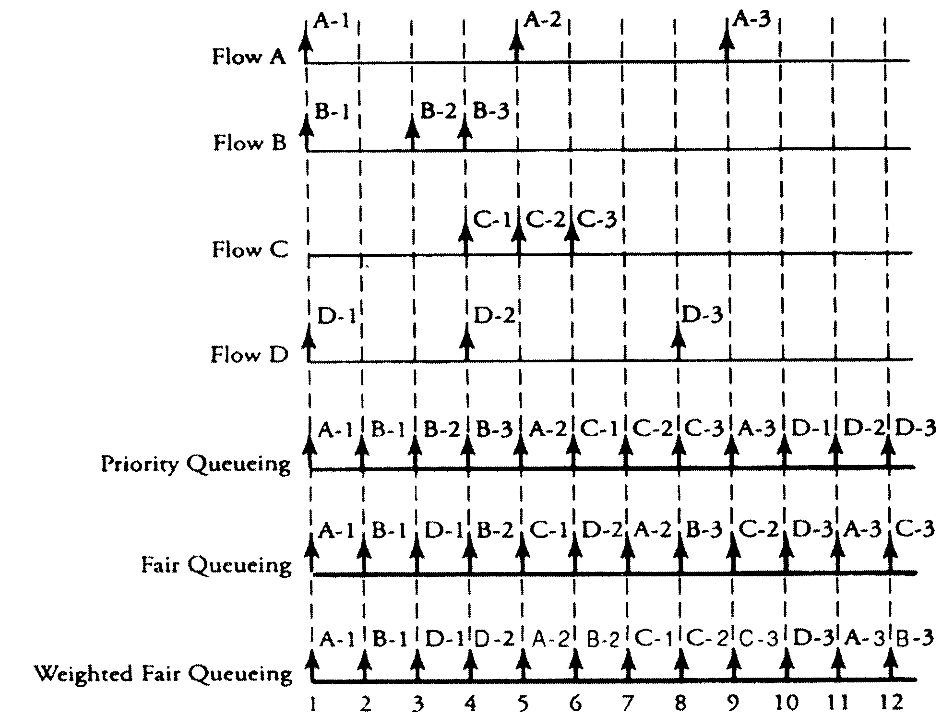

Flow A
A-1
A-2
A-3
Flow B
B-1
B-2
B-3
Flow C
C-1
C-2
C-3
Flow D
D-1
D-2
D-3
Priority Queueing
A-1 B-1 B-2 B-3 A-2 C-1 C-2 C-3 A-3 D-1 D-2 D-3
Fair Queueing
A-1 B-1 D-1 B-2 C-1 D-2 A-2 B-3 C-2 D-3 A-3 C-3
Weighted Fair Queueing
A-1 B-1 D-1 D-2 A-2 B-2 C-1 C-2 C-3 D-3 A-3 B-3
1 2 3 4 5 6 7 8 9 10 11 12

Figure 12.13 Comparison of priority queueing, fair queueing, and weighted fair queueing on processing of four flows

b_m is defined as the minimum bit-allocation value among all flows. Thus, we have $b_m = \min\{b_i\}$. In every cycle time, active flows are placed in an active list and serviced in round-robin order. If a packet cannot be completely serviced in a round of service without exceeding b_i, the packet is kept active until the next service round, and the unused portion of b_i is added to the next round. If a flow becomes inactive, the deficit is not accumulated to the next round. If a flow becomes inactive, the deficit is not accumulated to the next round of service and is eventually dropped from the node.

Although DRR is fair in terms of throughput, it lacks any reasonable delay bound. Another major disadvantage of this technique is that the delay bound for a flow with a small share of bandwidth can be very large. For example, consider a situation in which all flows are active. A low-weighted flow located at the end of the active list has to wait for all the other flows to be serviced before its turn, even if it is transmitting a minimum-sized packet.

Earliest Deadline First Scheduler

The *earliest deadline first* (EDF) scheduler computes the departure deadline for incoming packets and forms a *sorted deadline list* of packets to ensure the required transmission rate and maximum delay guarantees. The key lies in the assignment of the deadline so that the server provides a delay bound for those packets specified in the list. Thus, the deadline for a packet can be defined as

$$D = t_a + T_s, \tag{12.23}$$

where t_a is the expected arrival time of a packet at the server, and T_s is the delay guarantee of the server associated with the queue that the packet arrives from.

12.3 Differentiated Services QoS

The *differentiated services* (DS), or DiffServ, approach provides a simpler and more scalable QoS. DS minimizes the amount of storage needed in a router by processing traffic flows in an aggregate manner, moving all the complex procedures from the core to the edge of the network. A *traffic conditioner* is one of the main features of a DiffServ node to protect the DiffServ domain. As shown in Figure 12.14, the traffic conditioner includes four major components: meter, marker, shaper, and dropper. A *meter* measures the traffic to make sure that packets do not exceed their traffic profiles. A *marker* marks or unmarks packets in order to keep track of their situations in the DS node. A *shaper* delays any packet that is not compliant with the traffic profile. Finally, a *dropper* discards any packet that violates its traffic profile.

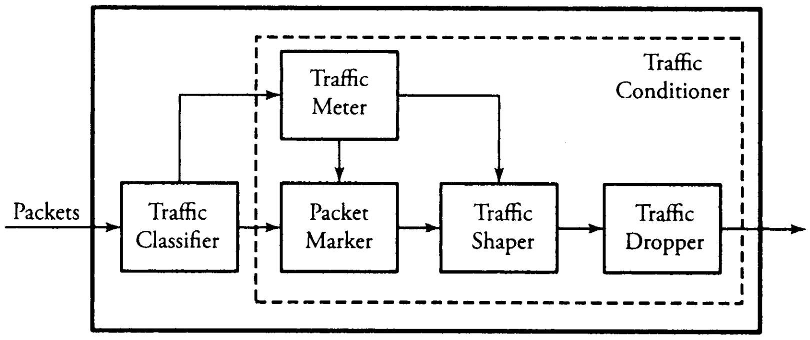


Figure 12.14 Overview of DiffServ operation

When users request a certain type of service, a service provider must satisfy the user's need. In order to allocate and control the available bandwidth within the DS domain, a bandwidth broker is needed to manage the traffic. The *bandwidth broker* operates in its own DS domain and maintains contact with other bandwidth brokers at neighboring domains. This is a way to confirm that a packet's requested service is valid throughout all domains within the routing path of the packet.

In order to process traffic flows in an aggregate manner, a packet must go through a *service-level agreements* (SLA) that includes a traffic-conditioning agreement (TCA). An SLA indicates the type of forwarding service, and a TCA presents all the detailed parameters that a customer receives. An SLA can be either static or dynamic. A static SLA is a long-term agreement, and a dynamic SLA uses the bandwidth broker that allows users to make changes more frequently. A user preferring packet-dependent quality-of-service can simply mark different values in the *type of service* (ToS) field at either the host or its access router. Routers in the DS model then detect the value in the DS field *in per hop behaviors* (PHBs). The quality-of-service can then be performed in accordance with the PHB.

In order to establish a traffic-policing scheme, a service provider uses a *traffic classifier* and a *traffic conditioner* at the domain's edge router when packets enter the service provider's network. The traffic classifier routes packets to specific outputs, based on the values found inside multiple fields of a packet header. The traffic conditioner detects and responds if any packet has violated any of the rules specified in the TCA. The DiffServ field value is set at the network boundaries. A DiffServ router uses the traffic classifier to select packets and then uses buffer management and a scheduling mechanism to deliver the specific PHB. The 8-bit DiffServ field is intended to replace the IPv4 ToS field and the IPv6 traffic class field. Six bits are used as a *differentiated services code point* (DSCP) to specify its PHB. The last 2 bits are unused and are ignored by the DS node.

12.3.1 Per-Hop Behavior (PHB)

We define two PHBs: *expedited forwarding* and *assured forwarding*. As for DiffServ domains, the expedited forwarding PHB provides low-loss, low-latency, low-jitter, ensured bandwidth and end-to-end services. Low latency and ensured bandwidth can be provided with a few configurations on the DiffServ node. Both the aggregate arrival rate for expedited-forwarding PHB packets and the aggregate arrival rate should be less than the aggregate minimum departure rate.

Several types of queue-scheduling mechanisms may be used to implement expedited-forwarding PHB. Ensured-forwarding PHB delivers packets with high assurance and high throughput, as long as the aggregate traffic does not exceed TCA. However, users are allowed to violate TCA, but the traffic beyond TCA is not given high assurance. Unlike the expedited forwarding PHB, the ensured-forwarding PHB does not provide low-latency and low-jitter application. The ensured forwarding PHB group can be classified into three service types: good, average, and poor. Three possible drop-precedence values are then assigned to packets within each class, determining the priority of the corresponding packet.

12.4 Resource Allocation

Scheduling algorithms provide QoS guarantees to traffic flows by controlling the flow of transmission, but sufficient buffer space has to be allocated to hold incoming packets, especially in high-speed networks. Some form of protection mechanism has to be implemented to provide flow isolation for preventing ill-behaving flows from occupying the entire queueing buffer. In addition, the mechanism must make packet-discard decisions on the basis of the network congestion level. Thus, buffer management is required to provide rate guarantees for a network.

Issues that involve *resource allocation* for packet-switched networks are fundamentally different from shared-access networks, such as Ethernet, where end hosts are able to directly observe the traffic on the shared medium. End hosts can thus tailor the rate at which they generate and send traffic, based on the traffic on the medium.

Consider the traffic-congestion case shown in Figure 12.15, where two links with a total traffic volume of $\alpha 1 + \alpha 2$ are connected to router R1 with an immediate outgoing link capacity α, where $\alpha < \alpha 1 + \alpha 2$. When it reaches the router, the sum of traffic encounters a low speed at the outgoing gate of the router. Router R1 starts to build up unserved packets and eventually enters a state of congestion. Situations like this, in

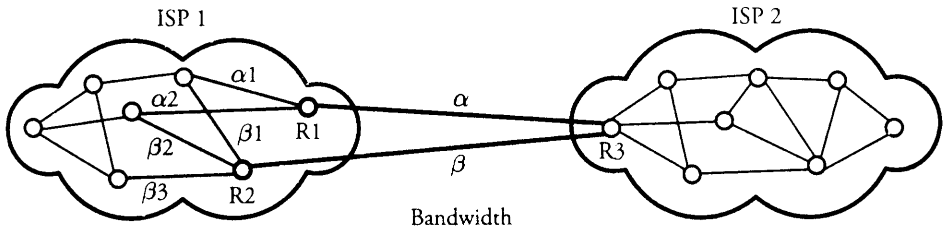


Figure 12.15 The need for network resource allocation when outgoing link capacity (α) is less than the sum of incoming ($\alpha 1 + \alpha 2$) traffic at a router

which traffic from several sources has to be combined onto a common low-speed link, are frequent on the Internet but are unusual in shared-access networks.

If the same situation occurs on router R2, where $\beta < \beta 1 + \beta 2 + \beta 3$, the frequency of severe congestion edge routers of the ISP 1 network is possible. This situation is by all means undesired from the standpoint of network management. Another important situation shown in Figure 12.15 is that router R3, located in a different domain, must be able to handle a great volume of traffic totaled at $\alpha + \beta$. This situation must convince a network manager that the type of router must depend on the location of the router in the network. The type of a router in a specific location of the network must be determined after thorough study, simulation, and taking all the necessary network design factors into consideration.

12.4.1 Management of Resources

Connection-oriented networks have a method of congestion control that leads to an underutilization of network resources. In the virtual-circuit approach, end hosts send a connection set-up message, which traverses the network and reserves buffers at each router for the connection. If adequate resources are not available at each router along the path of the connection, the connection is denied. The resources reserved for one connection cannot be used by another connection, leading to an underutilization of buffers at routers.

Service models of resource allocation are of two basic types:

1. *Best effort*. A network provides no guarantee to end hosts on the type of service to be delivered.
2. *QoS*. An end host may request a QoS connection. The host chooses from a set of QoS classes that the network supports.

In connectionless networks, datagrams are switched independently, and datagrams flowing through a router are called connectionless flows. The router maintains state of each flow to make necessary informed decisions. For example, in Figure 12.15, either router R2 or source host H2 can initiate a flow record. In some cases, the router observes the source/destination address combination for a packet and classifies packets into different flows. In some other cases, before a flow begins, the source sends some information to identify a connectionless flow originating from the host.

12.4.2 Classification of Resource-Allocation Schemes

Resource-allocation schemes can be classified as router based versus host based, fixed versus adaptive, and window based versus rate based.

Router Based versus Host Based

Resource allocation can be classified according to whether a router or a host sets up required resources. In a *router-based scheme*, routers have primary responsibility foı congestion control. A router selectively forwards packets or drops them, if required, to manage the allocation of existing resources.

A router also sends information to end hosts on the amount of traffic it can generate and send. In a *host-based scheme*, end hosts have primary responsibility for congestion control. Hosts observe the traffic conditions, such as throughput, delay, and packet losses, and adjust the rate at which they generate and send packets accordingly. In most networks, resource-allocation schemes may place a certain level of responsibility on both routers and end hosts.

Fixed versus Adaptive

In *fixed reservation schemes*, end hosts request resources at the router level before a flow begins. The router then allocates enough resources, such as bandwidth and buffer space, for the flow, based on its available resources. A router also ensures that the new reservation does not affect the quality of service provided to the existing reservations. If resources are unavailable, the router rejects the request from the end host. In an *adaptive reservation scheme*, end hosts send packets without reserving resources at the router and then adjust their sending rates, based on observable traffic conditions or the response from the router.

The observable traffic conditions are the amount of packet loss, delay, or other metrics. A router may also send messages to end hosts to slow down their sending rate. *Fixed reservation schemes* are router based, as the router is responsible for allocating sufficient resources for the flow. Thus, routers have primary responsibility for allocating and managing resources. Adaptive reservation schemes can be either router based or host based. If end hosts adjust rates of transmitted packets based on observable traffic conditions, the scheme is typically host based.

Window Based versus Rate Based

In *window-based resource allocation*, a receiver chooses a window size. This window size is dependent on the buffer space available to the receiver. The receiver then sends this window size to the sender. The sender transmits packets in accordance with the window size advertised by the receiver. In *rate-based resource allocation*, a receiver specifies a maximum rate of bits per second (b/s) it can handle. A sender sends traffic in compliance with the rate advertised by the receiver. The reservation-based allocation scheme might also involve reservations in b/s. In this case, routers along the path of a flow can handle traffic up to the advertised rate.

12.4.3 Fairness in Resource Allocation

The effectiveness of a resource-allocation scheme can be evaluated by considering two primary metrics: throughput and delay. Throughput has to be as large as possible; delay for a flow should normally be minimal. When the number of packets admitted into the network increases, the throughput tends to improve. But when the number of packets increases, the capacity of links is saturated, and thus the delay also increases, because a larger number of packets are buffered at the intermediate routers, resulting in increased delay.

A more effective method of evaluating the effectiveness of a resource-allocation scheme is to consider the ratio of throughput to delay, or *power*. As the number of packets admitted into a network builds up, the ratio of the throughput to delay increases. This is true until the network load threshold for low delay is reached. Beyond this limit, the network is overloaded, and the power drops rapidly.

A resource-allocation scheme must also be fair. This implies that each traffic flow through the network receives an equal share of the bandwidth. However, disregarding the flow throughputs (flow rates) itself is not fair. Raj Jain proposes a *fairness index* for n flows $f_1, f_2, \cdots, f_n$ as

$$\sigma = \frac{(\sum_{i=1}^{n} f_i)^2}{n \sum_{i=1}^{n} f_i^2}. \qquad (12.24)$$

The fairness index σ is always between 0 and 1 to represent the lowest and the best fairness. With reservation-based resource-allocation schemes, it is always possible for certain traffic, such as voice, to achieve a greater share of the available bandwidth by reservation. This may lead to unfairness in resource allocation. A fair resource-allocation scheme may not always be the most effective method of resource allocation.

12.4.4 ATM Resource Allocation

Traffic management and congestion control differ in ATM networks and regular packet-switched networks. The high-speed multiplexing and the small packet sizes in ATM networks make the task of congestion control difficult. One of the restrictions that ATM networks face in traffic control is an insufficient number of bits available in a cell for network control. Maintaining constant bit rate would help avoid the congestion in ATM networks.

Achieving Constant Bit Rate (CBR)

ATM networks are designed for fast cell switching and efficient routing at switches. One of the important requirements in ATM networks is low cell delay along with