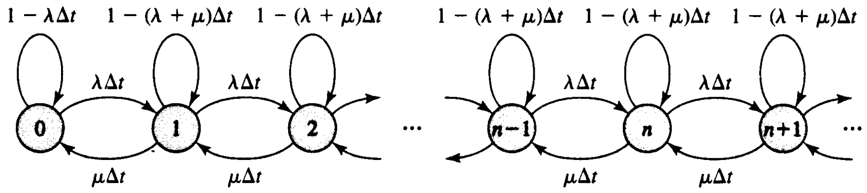


FIGURE A.10 State transition diagram.

Equation A.32 gives the probability that the number in the system increases by 1 in Δt seconds.

$$\begin{aligned} P[\text{no arrival \& 1 departure in } \Delta t] &= \{1 - \lambda\Delta t + o(\Delta t)\}\{\mu\Delta t + o(\Delta t)\} \\ &= \mu\Delta t + o(\Delta t). \end{aligned} \qquad \text{(A.33)}$$

Equation A.33 gives the probability that the number in a nonempty system decreases by 1 in Δt seconds. Note that the preceding equations imply that $N(t)$ always changes by single arrivals or single departures.

Figure A.10 shows the state transition diagram for $N(t)$, the number in the system. $N(t)$ increases by 1 in the next Δt seconds with probability $\lambda\Delta t$ and decreases by 1 in the next Δt seconds with probability $\mu\Delta t$. Note that every transition n to $n + 1$ cannot recur until the reverse transition $n + 1$ to n occurs. Therefore, if the system is stable, that is, if it does not grow steadily to infinity, then the long-term transition rate from n to $n + 1$ must equal the long-term transition rate from $n + 1$ to n.

Let p_n be the probability that n customers are in the system; then p_n is also the proportion of time that the system is in state n. Therefore, $p_n\lambda\Delta t$ is the transition from state n to state $n + 1$. Similarly, $p_{n+1}\mu\Delta t$ is the transition rate from $n + 1$ to n. This discussion implies that the two transition rates must be equal. Therefore

$$p_{n+1}\mu\Delta t = p_n\lambda\Delta t \qquad \text{(A.34)}$$

This implies that

$$p_{n+1} = (\lambda/\mu)p_n \quad n = 0, 1, \ldots, K \qquad \text{(A.35)}$$

Repeated applications of the preceding recursion imply that

$$p_{n+1} = (\lambda/\mu)^{n+1}p_0 = \rho^n p_0 \quad n = 0, 1, \ldots, K \qquad \text{(A.36)}$$

To find p_0, we use the fact that the probabilities must add up to 1:

$$\begin{aligned} 1 &= p_0 + p_1 + p_2 + \ldots + p_K = p_0\{1 + \rho + \rho^2 + \rho^3 + \ldots + \rho^K\} \\ &= p_0\frac{1 - \rho^{K+1}}{1 - \rho} \end{aligned} \qquad \text{(A.37)}$$

which implies that

$$p_0 = \frac{1 - \rho}{1 - \rho^{K+1}}. \qquad \text{(A.38)}$$

Finally we obtain the probabilities for the number of customers in the system:

$$P[N(t) = n] = p_n = \frac{(1-\rho)\rho^n}{1-\rho^{K+1}}, \quad \text{for } n = 0, 1, \ldots, K \tag{A.39}$$

The probability of blocking or loss in the M/M/1/K system is given by $P_{loss} = p_K$, which is the proportion of time that the system is full.

Consider what happens to the state probabilities as the load ρ is varied. For ρ less than 1, which corresponds to $\lambda < \mu$, the probabilities decrease exponentially as n increases; thus the number in the system tends to cluster around $n = 0$. In particular, adding more buffers is beneficial when $\lambda < \mu$, since the result is a reduction in loss probability. When $\rho = 1$, the normalization condition implies that all the states are equally probable; that is, $p_n = 1/(K+1)$. Once ρ is greater than 1, the probabilities actually increase with n and tend to cluster toward $n = K$; that is, the system tends to be full, as expected. Note that adding buffers when $\lambda > \mu$ is counterproductive, since the system will fill up the additional buffers. This result illustrates a key point in networking: The arrival rate should not be allowed to exceed the maximum capacity of a system for extended periods of time. The role of *congestion control* procedures in the network is to deal with this problem.

The average number of customers in the system $E[N]$ is given by

$$E[N] = \sum_{n=0}^{K} np_n = \sum_{n=0}^{K} n\frac{(1-\rho)\rho^n}{1-\rho^{K+1}} = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}} \tag{A.40}$$

Equation A.40 is valid for ρ not equal to 1. When $\rho = 1$, $E[N] = K/2$. By applying Little's formula, we obtain the average delay in an M/M/1/K system:

$$E[T] = \frac{E[N]}{\lambda(1-p_K)} \tag{A.41}$$

Now consider the M/M/1 system that has $K = \infty$. The state transition diagram for this system is the same as in Figure A.10 except that the states can assume all nonzero integer values. The probabilities are still related by

$$p_{n+1} = (\lambda/\mu)^{n+1}p_0 = \rho^n p_0 \quad n = 0, 1, \ldots \tag{A.42}$$

where $\rho = \lambda/\mu$. The normalization condition is now

$$\begin{aligned} 1 &= p_0 + p_1 + p_2 + \ldots = p_0\{1 + \rho + \rho^2 + \rho^3 + \ldots\} \\ &= p_0\frac{1}{1-\rho} \end{aligned} \tag{A.43}$$

Note that the preceding power series converges only if $\rho < 1$, which corresponds to $\lambda < \mu$. This result agrees with our intuition that an infinite-buffer system will be stable only if the arrival rate is less than the maximum departure rate. If not, the number in the system would grow without bound. We therefore find that the state probabilities are now

$$P\{N(t) = n\} = p_n = (1-\rho)\rho^n \quad n = 0, 1, \ldots (\rho < 1) \tag{A.44}$$

The average number in the system and the average delay are then given by

$$E[N] = \sum_{n=0}^{\infty} n(1-\rho)\rho^n = \frac{\rho}{(1-\rho)} \tag{A.45}$$

$$E[T] = \frac{E[N]}{\lambda} = \frac{1/\mu}{(1-\rho)} \tag{A.46}$$

The average waiting time is obtained as follows:

$$E[W] = E[T] - E[X] = \frac{(1/\mu)\rho}{(1-\rho)} \tag{A.47}$$

These three equations show that the average delay and the average waiting time grow without bound as ρ approaches 1. Thus we see that when arrivals or service times are random, perfect scheduling is not possible, so the system cannot be operated at $\lambda = \mu$.

A.3.2 Effect of Scale on Performance

The expressions for the M/M/1 system allow us to demonstrate the typical behavior of queueing systems as they are increased in scale. Consider a set of m separate M/M/1 systems, as shown in Figure A.11. Each system has an arrival rate of λ customers/second and a processing rate of μ customers/second. Now suppose that it is possible to combine the customer streams into a single stream with arrival rate $m\lambda$ customers/second. Also suppose that the processing capacities are combined into a single processor with rate $m\mu$ customers/second. The mean delay in the separate systems is given by

$$E[T_{separate}] = \frac{1/\mu}{(1-\rho)} \tag{A.48}$$

The combined system has an arrival rate $\lambda' = m\lambda$ and a processing rate $\mu' = m\mu$; therefore, its utilization is $\rho' = \lambda'/\mu' = \rho$. Therefore, the mean delay in the combined system is

$$E[T_{combined}] = \frac{1/\mu'}{(1-\rho)} = \frac{1/m\mu}{(1-\rho)} = \frac{1}{m}E[T_{separate}] \tag{A.49}$$

Thus we see that the combined system has a total delay of $1/m$ of the separate systems.

The improved performance of the combined system arises from improved global usage of the processors. In the separate systems some of the queues may be empty

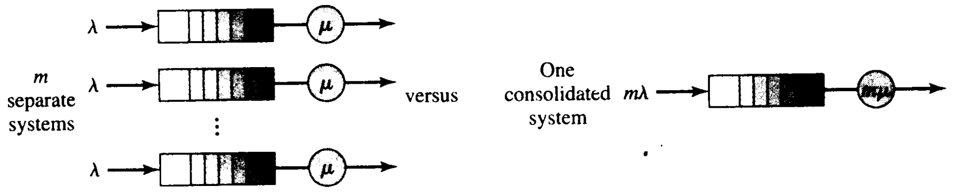


FIGURE A.11 Multiplexing gain and effect of scale.

while others are not. Consequently, some processors can be idle, even though there is work to be done in the system. In the combined system the processor will stay busy as long as customers are waiting to be served.

A.3.3 Average Packet Delay in a Network

In Section A.1.2, we used Little's formula to obtain an expression for the average delay experienced by a packet traversing a network. As shown in Figure A.5, the packet-switching network was modeled as many interconnected multiplexers. The average delay experienced by a packet in traversing the network is

$$E[T_{net}] = \frac{1}{\lambda_{net}} \sum_m \lambda_m E[T_m] \qquad \text{(A.50)}$$

where $E[T_m]$ is the average delay experienced in each multiplexer. To apply the results from our M/M/1 analysis, we need to make several assumptions. The most important assumption is that the service times that a packet experiences at different multiplexers are independent of each other. In fact, this assumption is untrue, since the service time is proportional to the packet length, and a packet has the same length as it traverses the network. Nevertheless, it has been found that this *independence assumption* can be used in larger networks.

If we model each multiplexer by an M/M/1 queue, we can then use the corresponding expression for the average delay:

$$E[T_m] = (1/\mu)(1 - \rho_m), \quad \text{where } \rho_m = \lambda_m/\mu \qquad \text{(A.51)}$$

where λ_m is the packet arrival rate at the mth multiplexer. The average packet delay in the network is then

$$E[T_m] = \sum_m \frac{1}{\lambda_{net}} \left(\frac{\rho_m}{1 - \rho_m} \right) \qquad \text{(A.52)}$$

This simple expression was first derived by [Kleinrock 1964]. It can be used as the basis for selecting the transmission speeds in a packet-switching network. It can also be used to synthesize routing algorithms that distribute the flows of packets over the network so as to keep the overall packet delay either minimum or within some range.

A.4 THE M/G/1 MODEL

The M/M/1 models derived in the previous sections are extremely useful in obtaining a quick insight into the trade-offs between the basic queueing systems parameters. The M/G/1 queueing model allows us to consider a more general class of resource-sharing systems. In particular, the service time can have any distribution and is not restricted to be exponential. The derivation of the M/G/1 results is beyond the scope of our discussion; we simply present the results and apply them to certain multiplexer

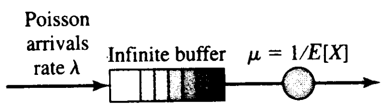


FIGURE A.12 M/G/1 queueing system has a general service time distribution.

problems. The reader is referred to [Leon-Garcia 1994] for a more detailed discussion of this model.

As shown in Figure A.12, the M/G/1 model assumes Poisson arrivals with rate λ, service times X with a general distribution $F_x(x) = P[X \leq x]$ that has a mean $E[X]$ and a variance $VAR[X]$, a single server, and unlimited buffer space.

The mean waiting time in an M/G/1 queueing system is given by

$$E[W] = \frac{\lambda E[X^2]}{2(1-\rho)} \tag{A.53}$$

Using the fact that $E[X^2] = VAR[X] + E[X]^2$, we obtain

$$E[W] = \frac{\lambda(VAR[X] + E[X]^2)}{2(1-\rho)} = \frac{\rho(1+C_X^2)}{2(1-\rho)}E[X] \tag{A.54}$$

where the coefficient of variation of the service time is given by $C_X^2 = VAR[X]/E[X]^2$.

The mean delay of the system is obtained by adding the mean service time to $E[W]$:

$$E[T] = E[W] + E[X] \tag{A.55}$$

The mean number in the system $E[N]$ and the mean number in queue $E[N_q]$ can then be found from Little's formula.

A.4.1 Service Time Variability and Delay

The mean waiting time in an M/G/1 system increases with the coefficient of variation of the service time. Figure A.13 shows the coefficient of variation for several service time distributions. The exponential service time has a coefficient of variation of 1 and serves as a basis for comparison. The constant service time has zero variance and hence has a coefficient of variation of zero. Consequently, its mean waiting time is one-half that of an M/M/1 system. The greater randomness in the service times of the M/M/1 system results in a larger average delay in the M/M/1 system.

Figure A.13 also shows two other types of service time distributions. The Erlang distribution has a coefficient of variation between 0 and 1, and the hyperexponential

	M/D/1	M/Er/1	M/M/1	M/H/1
Interarrivals	Constant	Erlang	Exponential	Hyperexponential
C_X^2	0	<1	1	>1
$E[W]/E[W_{M/M/1}]$	1/2	1/2<, <1	1	>1

FIGURE A.13 Comparison of mean waiting times in M/G/1 systems.

distribution has a coefficient of variation greater than 1. These two distributions can be used to model various degrees of randomness in the service time relative to the M/M/1 system.

In Chapter 6 we used the M/G/1 formula in assessing various types of schemes for sharing broadcast channels.

A.4.2 Priority Queueing Systems

The M/G/1 model can be generalized to the case where customers can belong to one of K priority classes. When a customer arrives at the system, the customer joins the queue of its priority class. Each time a customer service is completed, the next customer to be served is selected from the head of the line of the highest priority nonempty queue. We assume that once a customer begins service, it cannot be preempted by subsequent arrivals of higher-priority customers.

We will assume that the arrival at each priority class is Poisson with rate λ_k and that the average service time of a class k customer is $E[X_k]$, so the load offered by class k is $\rho_k = \lambda_k E[X_k]$.

It can be shown that if the total load is less than 1; that is

$$\rho = \rho_1 + \rho_2 + \ldots + \rho_K < 1 \tag{A.56}$$

then the average waiting time for a type k customer is given by

$$E[W_k] = \frac{\lambda E[X^2]}{(1 - \rho_1 - \rho_2 - \ldots - \rho_{k-1})(1 - \rho_1 - \rho_2 - \ldots - \rho_k)} \tag{A.57}$$

where

$$E[X^2] = \frac{\lambda_1}{\lambda} E[X_1^2] + \frac{\lambda_2}{\lambda} E[X_2^2] + \ldots + \frac{\lambda_K}{\lambda} E[X_K^2] \tag{A.58}$$

The mean delay is found by adding $E[X_k]$ to the corresponding waiting time. The interesting result in the preceding expression is in the terms in the denominator that indicate at what load a given class saturates. For example, the highest priority class has average waiting time

$$E[W_1] = \frac{\lambda E[X^2]}{(1 - \rho_1)} \tag{A.59}$$

so it saturates as ρ_1 approaches 1. Thus the saturation point of class 1 is determined only by its own load. On the other hand, the waiting time for class 2 is given by

$$E[W_2] = \frac{\lambda E[X^2]}{(1 - \rho_1)(1 - \rho_1 - \rho_2)} \tag{A.60}$$

The class 2 queue will saturate when $\rho_1 + \rho_2$ approaches 1. Thus the class 2 queue saturation point is affected by the class 1 load. Similarly, the class k queue saturation point depends on the sum of the loads of the classes of priority up to k. This result was used in Chapter 7 in the discussion of priority queueing disciplines in packet schedulers.

A.4.3 Vacation Models and Multiplexer Performance

The M/G/1 model with vacations arises in the following way. Consider an M/G/1 system in which the server goes on vacation (becomes unavailable) whenever it empties the queue. If upon returning from vacation, the server finds that the system is still empty, the server takes another vacation, and so on until it finds customers in the system. Suppose that vacation times are independent of each other and of the other variables in the system. If we let V be the vacation time, then the average waiting time in this system is

$$E[W] = \frac{\lambda E[X^2]}{2(1-\rho)} + \frac{E[V^2]}{2E[V]} \tag{A.61}$$

The M/G/1 vacation model is very useful in evaluating the performance of various multiplexing and medium access control systems. As a simple example consider an ATM multiplexer in which cells arrive according to a Poisson process and where cell transmission times are constrained to begin at integer multiples of the cell time. When the multiplexer empties the cell queue, then we can imagine that the multiplexer goes away on vacation for one cell time, just as modeled by the M/G/1 vacation model. If the cell transmission time is the constant X, then a vacation time is also $V = X$. Therefore, the average waiting time in this ATM multiplexer is

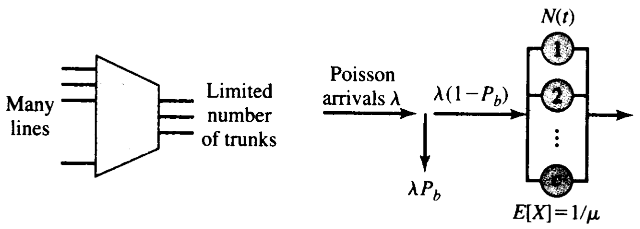
$$E[W] = \frac{\lambda X^2}{2(1-\rho)} + \frac{E[X^2]}{2E[X]} = \frac{\rho X}{2(1-\rho)} + \frac{X}{2} \tag{A.62}$$

The first term is the average waiting time in an ordinary M/G/1 system. The second term is the average time that elapses from the arrival instant of a random customer arrival to the beginning of the next cell transmission time.

A.5 ERLANG B FORMULA: M/M/c/c System

In Figure A.14 we show the M/M/c/c queueing model that can be used to model a system that handles trunk connection requests from many users. We assume trunk requests with exponential interarrival times with rate λ requests/second. Each trunk is viewed as a server, and the connection time is viewed as the service time X. Thus each trunk or server has a service rate $\mu = 1/E[X]$. We assume that connection requests are blocked if all the trunks are busy.

The state of the preceding system is given by $N(t)$, the number of trunks in use. Each new connection increases $N(t)$ by 1, and each connection release decreases $N(t)$ by 1. The state of the system then takes on the values $0, 1, \ldots, c$. The probability of a connection request in the next Δt seconds is given by $\lambda \Delta t$. The M/M/c/c queueing model differs from the M/M/1 queueing model in terms of the departure rate. If $N(t) = n$ servers are busy, then each server will complete its service in the next Δt seconds



- Blocked calls are cleared from the system; no waiting allowed.
- Performance parameter: P_b = fraction of arrivals that are blocked.
- $P_b = P[N(t) = c] = B(c, a)$ where $a = \lambda/\mu$.
- $B(c, a)$ is the Erlang B formula, which is valid for ***any*** service time distribution.

$$B(c, a) = \frac{\frac{a^c}{c!}}{\sum_{j=0}^{c} \frac{a^j}{j!}}$$

FIGURE A.14 M/M/c/c and the Erlang B formula.

with probability $\mu\Delta t$. The probability that one of the connections completes its service and that the other $n - 1$ continue their service in the next Δt seconds is given by

$$n(\mu\Delta t)(1 - \mu\Delta t)^{n-1} \approx n\mu\Delta t \tag{A.63}$$

The probability that two connections will complete their service is proportional to $(\mu\Delta t)^2$, which is negligible relative to Δt. Therefore, we find that the departure rate when $N(t) = n$ is $n\mu$.

Figure A.15 shows the state transition diagram for the M/M/c/c system. Proceeding as in the M/M/1/K analysis, we have

$$p_{n+1}(n + 1)\mu\Delta t = p_n\lambda\Delta t \tag{A.64}$$

This result implies that

$$p_{n+1} = \frac{\lambda}{(n + 1)\mu} p_n \quad n = 0, 1, \ldots, c \tag{A.65}$$

Repeated applications of the preceding recursion imply that

$$p_{n+1} = \frac{(\lambda/\mu)^{n+1}}{(n + 1)!} p_0 \quad n = 0, 1, \ldots, c \tag{A.66}$$

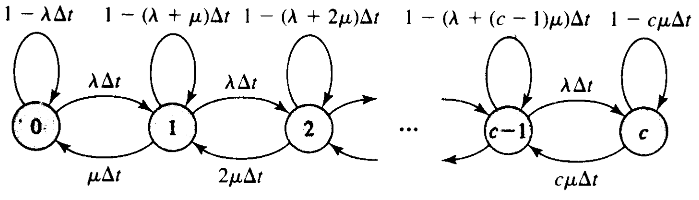


FIGURE A.15 State transition diagram for M/M/c/c.

Let the offered load be denoted by $a = \lambda/\mu$. To find p_0, we use the fact that the probabilities must add up to 1:

$$1 = p_0 + p_1 + p_2 + \cdots + p_c = p_0 \sum_{n=0}^{c} \frac{a^n}{n!} \tag{A.67}$$

Finally, we obtain the probabilities for the number of customers in the system:

$$P[N(t) = n] = p_n = \frac{\frac{a^n}{n!}}{\sum_{k=0}^{c} \frac{a^k}{k!}}, \quad \text{for } n = 0, 1, \ldots, c \tag{A.68}$$

The probability of blocking in the M/M/c/c system is given by $P_b = p_c$, which is the proportion of time that the system is full. This result leads to the Erlang B formula $B(c, a)$ that was used in Chapter 4.

Finally, we note that the Erlang B formula also applies to the M/G/c/c system; that is, the service time distribution need not be exponential.

FURTHER READING

Bertsekas, D. and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

Kleinrock, L., *Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill, New York, 1964.

Leon-Garcia. A., *Probability and Random Process for Electrical Engineering*, Addison-Wesley, Reading, Massachusetts, 1994.

See our website for additional references available through the internet.

APPENDIX B

Network Management

All networks, whether large or small, benefit from some form of management. Network management involves configuring, monitoring, and possibly reconfiguring components in a network with the goal of providing optimal performance, minimal downtime, proper security, accountability, and flexibility. This type of management is generally accomplished by using a *network management system*, which contains a software bundle designed to improve the overall performance and reliability of a system. In a small network, network management systems might be used to identify users who present security hazards or to find misconfigured systems. In a large network, network management systems might also be used to improve network performance and track resource usage for the purpose of accounting or charging.

As different types of networks have evolved over the years, so have different approaches to network management. The most common computer network management system currently implemented is the Simple Network Management Protocol (SNMP), which was originally intended to be a short-term solution to the network management issue. Alternatively, there is an OSI-based network management system called Common Management Information Protocol (CMIP). The OSI system was to be developed as the long-term solution. However, the ease with which SNMP was first implemented, as well as certain differences that emerged during the development of the OSI version, resulted in a continued separate development of SNMP.

The components that make up a network often come from many different vendors, and the operating systems that run the different systems in a network are often different. It is therefore important that a network management system be based on standards so that interoperability is also ensured. In this appendix we examine the concepts underlying network management in general and then present the most common approach that has been implemented for communication networks, namely, SNMP. Specifically we address the following topics:

1. *Network management overview.* We present the functions that can be performed by a network management system, and we describe the components that make up a network management system.
2. *SNMP.* We discuss the standards behind the most common network management protocol in use, the IETF-developed SNMP.
3. *Structure of Management Information.* We describe the Structure of Management Information (SMI), which defines the rules for describing management information.
4. *Management Information Base.* We describe the collection of objects, called the Management Information Base (MIB), that are managed by SNMP.
5. *Remote Network Monitoring.* We briefly discuss remote monitoring (RMON), which offers extensive network diagnostic, planning, and performance information.

B.1 NETWORK MANAGEMENT OVERVIEW

Network management involves monitoring and controlling a networking system so that it operates as intended. It also provides a means to configure the system while still meeting or exceeding design specifications. Note that management may be performed by a human, by an automated component, or both.

The functions performed by a network management system can be categorized into the following five areas:

Fault management refers to the detection, isolation, and resolution of network problems. Because a fault can cause part of a network to malfunction or even cause the whole network to fail completely, fault management provides a means for improving the reliability of the network. Examples include detecting a fault in a transmission link or network component, reconfiguring the network during the fault to maintain service level, and restoring the network when the fault is repaired.

Configuration management refers to the process of initially configuring a network and then adjusting it in response to changing network requirements. This function is perhaps the most important area of network management because improper configuration may cause the network to work suboptimally or to not work at all. An example is the configuration of various parameters on a network interface.

Accounting management involves tracking the usage of network resources. For example, one might monitor user load to determine how to better allocate resources. Alternatively, one might examine the type of traffic or the level of traffic that passes through a particular port. Accounting management also includes activities such as password administration and charging.

Performance management involves monitoring network utilization, end-to-end response time, and other performance measures at various points in a network. The results of the monitoring can be used to improve the performance of the network. Examples include tracking Ethernet utilization on all switched interfaces and reconfiguring new switched interfaces if performance is deemed to be below some specific level.

Security management refers to the process of making the network secure. This process, of course, involves managing the security services that pertain to access control, authentication, confidentiality, integrity, and nonrepudiation. Security is discussed in Chapter 11.

Although each specific network management architecture is based on different premises, certain concepts are common to most approaches to network management. A network contains a number of *managed devices* such as routers, bridges, switches, and hosts. Network management essentially involves monitoring and/or altering the configuration of such devices. An **agent** is a part of a network management system that resides in a managed device. The agent's tasks are to provide *management information* about the managed device and to accept instructions for configuring the device. It is also possible for an agent to not reside in the managed device; such an agent is called a *proxy agent.*

A **network management station** provides a text or graphical view of the entire network (or one of its components). This view is provided by way of a management application or manager that resides on the station. The manager exchanges management information with the agent by using a *network management protocol.* A management station allows a human or automated process to reconfigure a network, react to faults, observe performance, monitor security, and track usage—in short to implement the various network management functions required for that network.

More than one network management station can exist in a network. One network might have several stations, each of which provides a different view of the same portion of the network. Alternatively, a network might have several stations, each responsible for a different geographical portion of the network.

Figure B.1 shows a portion of a departmental network to illustrate how the network management concepts might apply. As shown in the figure, each host contains an agent that collects management information pertaining to the host. Similarly, the router also contains its own agent. The manager in the management station can poll a particular agent to obtain specific management information, which for example, can be the number of packet losses in the router. The management station can also alter some values in a managed device.

Note that a network management system may operate in a centralized or distributed manner or include both types of computing. In a centralized system one computer

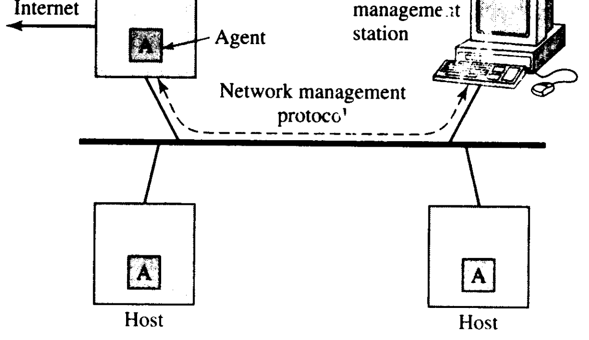


FIGURE B.1 A managed network.

system runs most of the applications required for network management. Similarly one database is located on the central machine. A distributed system may have several peer network management systems running simultaneously with each system managing a specific part of the network. The systems may be distributed geographically. Finally, a hierarchical network management system can have a centralized system at the root, with distributed peer systems running as children of the root.

B.2 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP)

In the early days of the Internet, the Internet Activities Board recognized the need for a management framework by which to manage TCP/IP implementations. The framework consists of three components:

1. A conceptual framework that defines the rules for describing management information, known as the Structure of Management Information (SMI).
2. A virtual database containing information about the managed device known as the Management Information Base (MIB).
3. A protocol for communication between a manager and an agent of a managed device, known as Simple Network Management Protocol (SNMP).

Essentially, the data that is handled by SNMP must follow the rules for objects in the MIB, which in turn are defined according to the SMI. These components are discussed separately below.

SNMP is an application layer protocol that is used to read and write variables in an agent's MIB. The most current version is SNMPv3. For convenience here SNMP refers to all versions of SNMP unless otherwise indicated.

SNMP is based on an asynchronous request-response protocol enhanced with trap-directed polling. The qualifier *asynchronous* refeis to the fact that the protocol need not wait for a response before sending other messages. *Trap-directed polling* refers to the fact that a manager polls in response to a trap message being sent by an agent, which occurs when there is an exception or after some measure has reached a certain threshold value. SNMP operates in a connectionless manner with UDP being the preferred transport mode. An SNMP manager sends messages to an agent via UDP destination port 161, while an agent sends trap messages to a manager via UDP destination port 162. The connectionless mode was chosen partly to simplify SNMP's implementation and because connectionless is usually the preferred mode by management applications that need to talk to many agents.

The messages (PDUs) exchanged via SNMP consist of a header and a data part. The header contains a version field, a community name field, and a PDU type field. The community name transmits a cleartext password between a manager and an agent, so this field can serve as a limited form of authentication.

SNMP provides three ways to access management information.

1. Request/response interaction in which a manager sends a request to an agent and the agent responds to the request. The request is usually to retrieve or modify

TABLE B.1 Request/Response interactions by role.

Role	Generate	Receive
Agent	`Response-PDU, Trap-PDU`	`GetRequest-PDU, GetNextRequest-PDU, GetBulkRequest-PDU, SetRequest-PDU, InformRequest-PDU`
Manager	`GetRequest-PDU, GetNextRequest-PDU GetBulkRequest-PDU, SetRequest-PDU, InformRequest-PDU`	`Response-PDU, Trap-PDU`

management information associated with the network device in question. Specific information is requested by a manager, using one of the following requests:

- `GetRequest-PDU` for requesting information on specific variables.
- `GetNextRequest-PDU` for requesting the next set of information (usually from a management information table).
- `GetBulkRequest-PDU` for requesting bulk information retrieval. This request was introduced in SNMPv2 to allow the retrieval of as much information as possible in a packet.
- `SetRequest-PDU` for creating or modifying management information.

The agent must always reply using a `Response-PDU`.

2. Request/response interaction in which a manager sends a request to another manager and the latter responds to the request. The request is usually to notify a manager of management information associated with the manager, using `InformRequest-PDU`.
3. Unconfirmed interaction in which an agent sends an unsolicited `Trap-PDU` to a manager. This request is usually to notify the manager of an exceptional situation that has resulted in changes to the management information associated with the network device.

Table B.1 shows another way of looking at the manager/agent roles and the functions they can perform.

A typical interaction between a manager and agent would proceed as follows. The manager issues some form of get request that contains a unique request-id to match the response with the request, a zero-valued error status/error index, and one or more variable bindings. The agent issues a response containing the *same* request-id, a zero-valued error status if there is no error, and the *same* variable bindings. Figure B.2 shows a typical interaction for an information request.

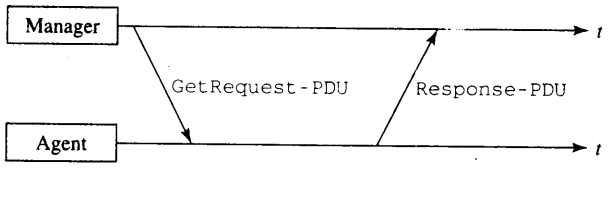


FIGURE B.2 Typical request/response interaction.

If an exception occurs for one or more of the variables, then the particular error status for each relevant variable is returned as well. For example, if the agent does not implement a particular variable, then the `noSuchName` error status is returned. Other error status values are `tooBig`, indicating that the response is too large to send; `badValue`, indicating that the set operation specified an invalid value or syntax; `readOnly`, indicating that a manager attempted to write a read-only variable; and `genErr` for some other errors.

Version 3 of SNMP was formally documented in early 1998 [RFC 2271]. It presents a more complex framework for message exchange, the complexity being required both for extensibility and for security reasons. The security system contains a user-based security model, as well as other security models that may be implemented. This model is intended to protect against the unauthorized modification of information of SNMP messages in transit, masquerading, eavesdropping, or deliberate tampering with the message stream (that is, a deliberate reordering, delay, or replay of messages). It does not protect against denial of service or unauthorized traffic analysis. The model uses the MD5 encryption scheme for verifying user keys, a SHA message digest algorithm (HMAC-SHA-96) to verify message integrity and to verify the user on whose behalf the message was generated, and a CBC-DES symmetric encryption protocol for privacy. See [RFC 2274] for further information on the user-based security model.

B.3 STRUCTURE OF MANAGEMENT INFORMATION

The **Structure of Management Information (SMI)** defines the rules for describing managed objects. In the SNMP framework managed objects reside in a virtual database called the Management Information Base (MIB). Collections of related objects are defined in MIB modules. The modules are written using a subset of **Abstract Syntax Notation One (ASN.1)**, which describes the data structures in a machine-dependent language. SNMP uses the **Basic Encoding Rule (BER)** to transmit the data structures across the network unambiguously.

Several data types are allowed in SMI. The primitive data types consist of `INTEGER`, `OCTET STRING`, `NULL`, and `OBJECT IDENTIFIER`. Additional user-defined data types are application specific. Primitive data types are written in uppercase, while user-defined data types start with an uppercase letter but contain at least one character other than an uppercase letter. Table B.2 lists some of the data types permitted in SMI.

An `OBJECT IDENTIFIER` is represented as a sequence of nonnegative integers where each integer corresponds to a particular node in the tree. This data type provides a means for identifying a managed object and relating its place in the object hierarchy. Figure B.3 shows the object identifier tree as it has been defined for various internet objects. A *label* is a pairing of a text description with an integer for a particular node, also called a subidentifier. The root node is unlabeled. Similarly, the integers that make up an object identifier are separated by periods (.). For example, as shown by the tree, the object identifiers for all internet objects start with 1.3.6.1.

TABLE B.2 SMI data types.

Data type	Description
`INTEGER`	A 32-bit integer
`OCTET STRING`	A string of zero or more bytes with each byte having a value between 0 to 255
`Display STRING`	A string of zero or more bytes with each byte being a character from the NVT ASCII set
`NULL`	A variable with no value
`OBJECT IDENTIFIER`	An authoritatively defined data type described below
`IpAddress`	A 32-bit Internet address represented as an octet string of length 4
`Counter`	A nonnegative integer that increases from 0 to $2^{32} - 1$ and then wraps back to 0
`Gauge`	A nonnegative integer that can increase or decrease, but which latches at a maximum value
`TimeTicks`	A nonnegative integer that counts the time in hundredths of a second since some epoch
`Opaque`	An opaquely encoded data string

The internet (1) subtree itself has six subtrees:

- The directory (1) subtree is reserved for future use describing how OSI directory may be used in the Internet.
- The mgmt (2) subtree is used to identify "standard" objects that are registered by the Internet Assigned Numbers Authority (IANA).
- The experimental (3) subtree is for objects being used experimentally by working groups of the IETF. If the object becomes a standard, then it must move to the mgmt (2) subtree.
- The private (4) subtree is for objects defined by a single party, usually a vendor. It has a subtree enterprise (1), which allows companies to register their network objects.
-

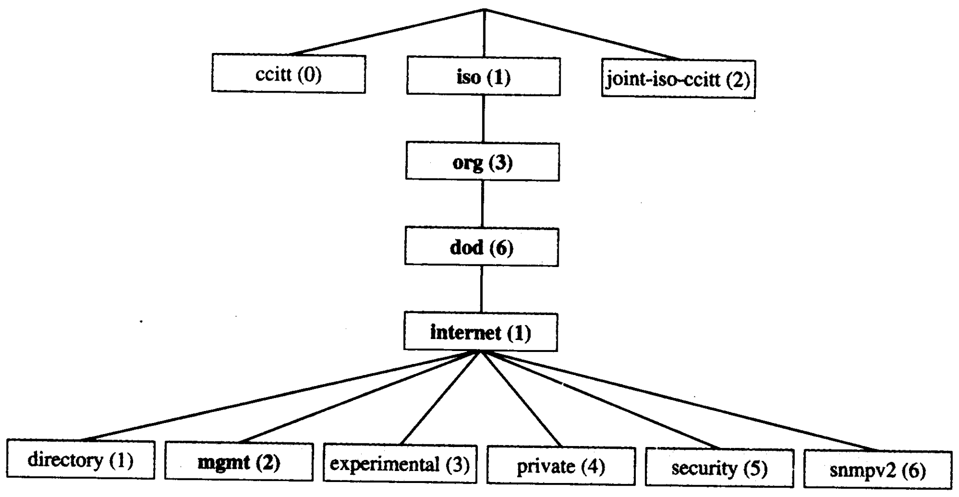


FIGURE B.3 Object identifier tree for internet (1) objects.