Principles of Computer System Design

An Introduction

Glossary

Jerome H. Saltzer

M. Frans Kaashoek

Massachusetts Institute of Technology

Version 5.0

Suggestions, Comments, Corrections, and Requests to waive license restrictions: Please send correspondence by electronic mail to:

Saltzer@mit.edu

and

kaashoek@mit.edu

Glossary

abort—Upon deciding that an all-or-nothing action cannot or should not commit, to undo all of the changes previously made by that all-or-nothing action. After aborting, the state of the system, as viewed by anyone above the layer that implements the all-or-nothing action, is as if the all-or-nothing action never existed. Compare with *commit*. [Ch. 9]

absolute path name—In a naming hierarchy, a path name that a name resolver resolves by using a universal context known as the *root* context. [Ch. 2]

abstraction—The separation of the interface specification of a module from its internal implementation so that one can understand and make use of that module with no need to know how it is implemented internally. [Ch. 1]

access control list (**ACL**)—A list of principals authorized to have access to some object. [Ch. 11]

acknowledgment (**ACK**)—A status report from the recipient of a communication to the originator. Depending on the protocol, an acknowledgment may imply or explicitly state any of several things, for example, that the communication was received, that its checksum verified correctly, that delivery to a higher level was successful, or that buffer space is available for another communication. Compare with *negative acknowledgment*. [Ch. 2]

action—An operation performed by an interpreter. Examples include a microcode step, a machine instruction, a higher-level language instruction, a procedure invocation, a shell command line, a response to a gesture at a graphical interface, or a database update. [Ch. 9]

active fault—A fault that is currently causing an error. Compare with *latent fault*. [Ch. 8]

adaptive routing—A method for setting up forwarding tables so that they change automatically when links are added to and deleted from the network or when congestion makes a path less desirable. Compare with *static routing*. [Ch. 7]

address—A name that is overloaded with information useful for locating the named object. In a computer system, an address is usually of fixed length and resolved by hardware into a physical location by mapping to geometric coordinates. Examples of addresses include the names for a byte of memory and for a disk track. Also see *network address*. [Ch. 2]

address resolution protocol (ARP)—A protocol used when a broadcast network is a component of a packet-forwarding network. The protocol dynamically constructs tables that map station identifiers of the broadcast network to network attachment point identifiers of the packet-forwarding network. [Ch. 7]

address space—The name space of a location-addressed memory, usually a set of contiguous integers (0, 1, 2,…). [Ch. 2]

adversary—An entity that intentionally tries to defeat the security measures of a computer system. The entity may be malicious, out for profit, or just a hacker. A friendly adversary is one that tests the security of a computer system. [Ch. 11]

advertise—In a network-layer routing protocol, for a participant to tell other participants which network addresses it knows how to reach. [Ch. 7]

alias—One of multiple names that map to the same value; another term for *synonym*. (Beware: some operating systems define *alias* to mean an *indirect name*.) [Ch. 2]

all-or-nothing atomicity—A property of a multistep action that if an anticipated failure occurs during the steps of the action, the effect of the action from the point of view of its invoker is either never to have started or else to have been accomplished completely. Compare with *before-or-after atomicity* and *atomic*. [Ch. 9]

any-to-any connection—A desirable property of a communication network, that any node be able to communicate with any other. [Ch. 7]

archive—A record, usually kept in the form of a log, of old data values, for auditing, recovery from application mistakes, or historical interest. [Ch. 9]

asynchronous (From Greek roots meaning "not timed")—1. Describes concurrent activities that are not coordinated by a common clock and thus may make progress at different rates. For example, multiple processors are usually asynchronous, and I/O operations are typically performed by an I/O channel processor that is asynchronous with respect to the processor that initiated the I/O. [Ch. 2] 2. In a communication network, describes a communication link over which data is sent in frames whose timing relative to other frames is unpredictable and whose lengths may not be uniform. Compare with *isochronous*. [Ch. 7]

at-least-once—A protocol assurance that the intended operation or message delivery was performed at least one time. It may have been performed several times. [Ch. 4]

at-most-once—A protocol assurance that the intended operation or message delivery was performed no more than one time. It may not have been performed at all. [Ch. 4]

atomic (adj.); **atomicity** (n.)—A property of a multistep action that there be no evidence that it is composite above the layer that implements it. An atomic action can be before-or-after, which means that its effect is as if it occurred either completely before or completely after any other before-or-after action. An atomic action can also be all-or-nothing, which means that if an anticipated failure occurs during the action, the effect of the action as seen by higher layers is either never to have started or else to have completed successfully. An atomic action that is *both* all-or-nothing and before-or-after is known as a *transaction*. [Ch. 9]

atomic storage—Cell storage for which a multicell PUT can have only two possible outcomes: (1) it stores all data successfully, or (2) it does not change the previous data at all. In consequence, either a concurrent thread or (following a failure) a later thread doing a GET will always read either all old data or all new data. Computer architectures in which multicell PUTs are not atomic are said to be subject to *write tearing*. [Ch. 9]

authentication—Verifying the identity of a principal or the authenticity of a message. [Ch. 11]

authentication tag—A cryptographically computed string, associated with a message, that allows a receiver to verify the authenticity of the message. [Ch. 11]

automatic rate adaptation—A technique by which a sender automatically adjusts the rate at which it introduces packets into a network to match the maximum rate that the narrowest bottleneck can handle. [Ch. 7]

authorization—A decision made by an authority to grant a principal permission to perform some operation, such as reading certain information. [Ch. 11]

availability—A measure of the time that a system was actually usable, as a fraction of the time that it was intended to be usable. Compare with its complement, *down time*. [Ch. 8]

backup copy—Of a set of replicas that is not written or updated synchronously, one that is written later. Compare with *primary copy* and *mirror*. [Ch. 10]

backward error correction—A technique for correcting errors in which the source of the data or control signal applies enough redundancy to allow errors to be detected and, if an error does occur, that source is asked to redo the calculation or repeat the transmission. Compare with *forward error correction*. [Ch. 8]

bad-news diode—An undesirable tendency of people in organizations that design and implement systems: good news, for example, that a module is ready for delivery ahead of schedule, tends to be passed immediately throughout the organization, but bad news, for example, that a module did not pass its acceptance tests, tends to be held locally until either the problem can be fixed or it cannot be concealed any longer. [Ch. 1]

bandwidth—A measure of analog spectrum spacefor a communication channel. The bandwidth, the acceptable signal power, and the noise level of a channel together determine the maximum possible data rate for that channel. In digital systems, this term is so often misused as a synonym for maximum data rate that it has now entered the vocabulary of digital designers with that additional meaning. Analog engineers, however, still cringe at that usage. [Ch. 7]

batching—A technique to improve performance by combining several operations into a single operation to reduce setup overhead. [Ch. 6]

before-or-after atomicity—A property of concurrent actions: Concurrent actions are before-or-after actions if their effect from the point of view of their invokers is the same as if the actions occurred either completely before or completely after one another. One consequence is that concurrent before-or-after software actions cannot discover the composite nature of one another (that is, one action cannot tell that another has multiple steps). A consequence in the case of hardware is that concurrent before-or-after WRITEs to the same memory cell will be performed in some order, so there is no danger that the cell will end up containing, for example, the OR of several WRITE values. The database literature uses the words "isolation" and "serializable", the operating system literature

uses the words "mutual exclusion" and "critical section", and the computer architecture literature uses the unqualified word "atomicity" for this concept. [Ch. 5] Compare with *all-or-nothing atomicity* and *atomic*. [Ch. 9]

best-effort contract—The promise given by a forwarding network when it accepts a packet: it will use its best effort to deliver the packet, but the time to delivery is not fixed, the order of delivery relative to other packets sent to the same destination is unpredictable, and the packet may be duplicated or lost. [Ch. 7]

binding (n.); **bind** (v.)—As used in naming, a mapping from a specified name to a particular value in a specified context. When a binding exists, the name is said to be **bound**. Binding may occur at any time up to and including the instant that a name is resolved. The term is also used more generally, meaning to choose a specific lower-layer implementation for some higher-layer feature. [Ch. 2]

bit error rate—In a digital transmission system, the rate at which bits that have incorrect values arrive at the receiver, expressed as a fraction of the bits transmitted, for example, one in 10^{10}. [Ch. 7]

bit stuffing—The technique of inserting a bit pattern as a marker in a stream of bits and then inserting bits elsewhere in the stream to ensure that payload data never matches the marker bit pattern. [Ch. 7]

blind write—An update to a data value X by a transaction that did not previously read X. [Ch. 9]

bootstrapping—A systematic approach to solving a general problem, consisting of a method for reducing the general problem to a specialized instance of the same problem and a method for solving the specialized instance. [Ch. 5]

bottleneck—The stage in a multistage pipeline that takes longer to perform its task than any of the other stages. [Ch. 6]

broadcast—To send a packet that is intended to be received by many (ideally, all) of the stations of a broadcast link (link-layer broadcast), or all the destination addresses of a network (network-layer broadcast). [Ch. 7]

burst—A batch of related bits that is irregular in size and timing relative to other such batches. Bursts of data are the usual content of messages and the usual payload of packets. One can also have bursts of noise and bursts of packets. [Ch. 7]

Byzantine fault—A fault that generates inconsistent errors (perhaps maliciously) that can confuse or disrupt fault tolerance or security mechanisms. [Ch. 8]

cache—A performance-enhancing module that remembers the result of an expensive computation on the chance that the result may soon be needed again. [Ch. 2]

cache coherence—Read/write coherence for a multilevel memory system that has a cache. It is a specification that the cache provide strict consistency at its interface. [Ch. 10]

capability—In a computer system, an unforgeable ticket, which when presented is taken as incontestable proof that the presenter is authorized to have access to the object named

in the ticket. [Ch. 11]

capacity—Any consistent measure of the size or amount of a resource. [Ch. 6]

cell storage—Storage in which a WRITE or PUT operates by overwriting, thus destroying previously stored information. Many physical storage devices, including magnetic disk and CMOS random access memory, implement cell storage. Compare with *journal storage*. [Ch. 9]

certificate—A message that attests the binding of a principal identifier to a cryptographic key. [Ch. 11]

certificate authority (CA)—A principal that issues and signs certificates. [Ch. 11]

certify—To check the accuracy, correctness, and completeness of a security mechanism. [Ch. 11]

checkpoint—1. (n.) Information written to non-volatile storage that is intended to speed up recovery from a crash. 2 (v.) To write a checkpoint. [Ch. 9]

checksum—A stylized error-detection code in which the data is unchanged from its uncoded form and additional, redundant data is placed in a distinct, separately architected field. [Ch. 7]

cipher—Synonym for a *cryptographic transformation*. [Ch. 11]

ciphertext—The result of encryption. Compare with *plaintext*. [Ch. 11]

circuit switch—A device with many electrical circuits coming in to it that can connect any circuit to any other circuit; it may be able to perform many such connections simultaneously. Historically, telephone systems were constructed of circuit switches. [Ch. 7]

cleartext—Synonym for *plaintext*. [Ch. 11]

client—A module that initiates actions, such as sending a request to a service. [Ch. 4] At the end-to-end layer of a network, the end that initiates actions. Compare with **service**. [Ch. 7]

client/service organization—An organization that enforces modularity among modules of a computer system by limiting the interaction among the modules to messages. [Ch. 4]

close-to-open consistency—A consistency model for file operations. When a thread opens a file and performs several write operations, all of the modifications weill be visible to concurrent threads only after the first thread closes the file. [Ch. 4]

closure—In a programming language, an object that consists of a reference to the text of a procedure and a reference to the context in which the program interpreter is to resolve the variables of the procedure. [Ch. 2]

coheerence—See *read/write coherence* or *cache coherence*.

collision—1. In naming, a particular kind of name conflict in which an algorithmic name generator accidentally generates the same name more than once in what is intended to be a unique identifier name space. [Ch. 3] 2. In networks, an event when two stations

attempt to send a message over the same physical medium at the same time. See also *Ethernet*. [Ch. 7]

commit—To renounce the ability to abandon an all-or-nothing action unilaterally. One usually commits an all-or-nothing action before making its results available to concurrent or later all-or-nothing actions. Before committing, the all-or-nothing action can be abandoned and one can pretend that it had never been undertaken. After committing, the all-or-nothing action must be able to complete. A committed all-or-nothing action cannot be abandoned; if it can be determined precisely how far its results have propagated, it may be possible to reverse some or all of its effects by compensation. Commitment also usually includes an expectation that the results preserve any appropriate invariants and will be durable to the extent that the application requires those properties. Compare with *compensate* and *abort*. [Ch. 9]

communication link—a data communication path between physically separated components. [Ch. 2]

compensate (adj.); **compensation** (n.)—To perform an action that reverses the effect of some previously committed action. Compensation is intrinsically application dependent; it is easier to reverse an incorrect accounting entry than it is to undrill an unwanted hole. [Ch. 9]

complexity—A loosely defined notion that a system has so many components, interconnections, and irregularities that it is difficult to understand, implement, and maintain. [Ch. 1]

confidentiality—Limiting information access to authorized principals. *Secrecy* is a synonym. [Ch. 11]

confinement—Allowing a potentially untrusted program to have access to data, while ensuring that the program cannot release information. [Ch. 11]

congestion—Overload of a resource that persists for significantly longer than the average service time of the resource. (Since significance is in the eye of the beholder, the concept is not a precise one.) [Ch. 7]

congestion collapse—When an increase in offered load causes a catastrophic decrease in useful work accomplished. [Ch. 7]

connection—A communication path that requires maintaining state between successive messages. See *set up* and *tear down.* [Ch. 7]

connectionless—Describes a communication path that does not require coordinated state and can be used without set up or tear down. See *connection*. [Ch. 7]

consensus—Agreement at separated sites on a data value despite communication failures. [Ch. 10]

consistency—A particular constraint on the memory model of a storage system that allows concurrency and uses replicas: that all readers see the same result. Also used in some professional literature as a synonym for *coherence*. [Ch. 10]

constraint—An application-defined invariant on a set of data values or externally visible actions. Example: a requirement that the balances of all the accounts of a bank sum to zero, or a requirement that a majority of the copies of a set of data be identical. [Ch. 10]

context—One of the inputs required by a name-mapping algorithm in order to resolve a name. A common form for a context is a set of name-to-value bindings. [Ch. 2]

context reference—The name of a context. [Ch. 2]

continuous operation—An availability goal, that a system be capable of running indefinitely. The primary requirement of continuous operation is that it must be possible to perform repair and maintenance without stopping the system. [Ch. 8]

control point—An entity that can adjust the capacity of a limited resource or change the load that a source offers. [Ch. 7]

cooperative scheduling—A style of thread scheduling in which each thread on its own initiative releases the processor periodically to allow other threads to run. [Ch. 5]

covert channel—In a flow-control security system, a way of leaking information into or out of a secure area. For example, a program with access to a secret might touch several shared but normally unused virtual memory pages in a pattern to bring them into real memory; a conspirator outside the secure area may be able to detect the pattern by measuring the time required to read those same shared pages. [Ch. 11]

cryptographic hash function—A cryptographic function that maps messages to short values in such a way that it is difficult to (1) reconstruct a message from its hash value; and (2) construct two different messages having the same value. [Ch. 11]

cryptographic key—The easily changeable component of a key-driven cryptographic transformation. A cryptographic key is a string of bits. The bits may be generated randomly, or they may be a transformed version of a password. The cryptographic key, or at least part of it, usually must be kept secret, while all other components of the transformation can be made public. [Ch. 11]

cryptographic transformation—Mathematical transformation used as a building block for implementing security primitives. Such building blocks include functions for implementing encryption and decryption, creating and verifying authentication tags, cryptographic hashes, and pseudorandom number generators. [Ch. 11]

cryptography—A discipline of theoretical computer science that specializes in the study of cryptographic transformations and protocols. [Ch. 11]

cut-through—A forwarding technique in which transmission of a packet or frame on an outgoing link begins while the packet or frame is still being received on the incoming link. [Ch. 7]

dallying—A technique to improve performance by delaying a request on the chance that the operation won't be needed, or to create more opportunities for batching. [Ch. 6]

dangling reference—Use of a name that has outlived the binding of that name. [Ch. 3]

data integrity—Authenticity of the apparent content of a message or file. [Ch. 11] In a

network, a transport protocol assurance that the data delivered to the recipient is identical to the original data the sender provided. Compare with *origin authenticity*. [Ch. 7]

data rate—The rate, usually measured in bits per second, at which bits are sent over a communication link. When talking of the data rate of an asynchronous communication link, the term is often used to mean the maximum data rate that the link allows. [Ch. 7]

deadlock—Undesirable interaction among a group of threads in which each thread is waiting for some other thread in the group to make progress. [Ch. 5]

decay—Unintended loss of stored state with the passage of time. [Ch. 2]

decay set—A set of storage blocks, words, tracks, or other physical groupings, in which all members of the set may spontaneously fail together, but independently of any other decay set. [Ch. 8]

decrypt—To perform a reverse cryptographic transformation on a previously encrypted message to obtain the plaintext. Compare with *encrypt*. [Ch. 11]

default context reference—A context reference chosen by the name resolver rather than specified as part of the name or by the object that used the name. Compare with *explicit context reference*. [Ch. 2]

demand paging—A class of page-movement algorithm that moves pages into the primary device only at the instant that they are used. Compare with *prepaging*. [Ch. 6]

destination—The network attachment point to which the payload of a packet is to be delivered. Sometimes used as shorthand for *destination address*. [Ch. 7]

destination address—An identifier of the destination of a packet, usually carried as a field in the header of the packet. [Ch. 7]

detectable error—An error or class of errors for which a reliable detection plan can be devised. An error that is not detectable usually leads to a failure, unless some mechanism that is intended to mask some other error accidentally happens to mask the undetectable error. Compare with *maskable error* and *tolerated error*. [Ch. 8]

digital signature—An authentication tag computed with public-key cryptography. [Ch. 11]

directory—In a file system, an object consisting of a table of bindings between symbolic file names and some description (e.g., a file number or a file map) of the corresponding file. Other terms used for this concept include *catalog* and *folder*. A directory is an example of a context. [Ch. 2]

discretionary access control—A property of an access control system. In a discretionary access control system, the owner of an object has the authority to decide which principals have access to that object. Compare with *non-discretionary access control*. [Ch. 11]

do action—(n.) Term used in some systems for a *redo action*. [Ch. 9]

domain—A range of addresses to which a thread has access. It is the abstraction that enforces modularity within a memory, separating modules and allowing for controlled

sharing. [Ch. 5]

down time—A measure of the time that a system was not usable, as a fraction of the time that it was intended to be usable. Compare with its complement, *availability*. [Ch. 8]

duplex—Describes a link or connection between two stations that can be used in both directions. Compare with *simplex*, *half-duplex,* and *full-duplex*. [Ch. 7]

duplicate suppression—A transport protocol mechanism for achieving at-most-once delivery assurance, by identifying and discarding extra copies of packets or messages. [Ch. 7]

durability—A property of a storage medium that, once written, it can be read for as long as the application requires. Compare with *stability* and *persistence*, terms that have different technical definitions as explained in Sidebar 2.7. [Ch. 2]

durable storage—Storage with the property that it (ideally) is decay-free, so it never fails to return on a GET the data that was stored by a previously successful PUT. Since that ideal is impossibly strict, in practice, storage is considered durable when the probability of failure is sufficiently low that the application can tolerate it. Durability is thus an application-defined specification of how long the results of an action, once completed, must be preserved. Durable is distinct from *non-volatile*, which describes storage that maintains its memory while the power is off, but may still have an intolerable probability of decay. The term *persistent* is sometimes used as a synonym for durable, as explained in Sidebar 2.7, but to minimize confusion this text avoids that usage. [Ch. 8]

dynamic scope—An example of a default context, used to resolve names of program variables in some programming languages. The name resolver searches backward in the call stack for a binding, starting with the stack frame of the procedure that used the name, then the stack of its caller, then the caller's caller, and so on. Compare with *static scope*. [Ch. 2]

earliest deadline first scheduling policy—A scheduling policy for real-time systems that gives priority to the thread with the earliest deadline. [Ch. 6]

early drop—A predictive strategy for managing an overloaded resource: the system refuses service to some customers before the queue is full. [Ch. 7]

emergent property—A property of an assemblage of components that would not be predicted by examining the components individually. Emergent properties are a surprise when first encountered. [Ch. 1]

emulation—Faithfully simulating some physical hardware so that the simulated hardware can run any software that the physical hardware can. [Ch. 5]

encrypt—To perform a cryptographic transformation on a message with the objective of achieving confidentiality. The cryptographic transformation is usually key-driven. Compare with the inverse operation, **decrypt**, which can recover the original message. [Ch. 11]

end-to-end—Describes communication between network attachment points, as contrasted with communication between points within the network or across a single

link. [Ch. 7]

end-to-end layer—The communication system layer that manages end-to-end communications. [Ch. 7]

enforced modularity—Modularity that prevents accidental errors from propagating from one module to another. Compare with **soft modularity**. [Ch. 4]

enumerate—To generate a list of all the names that can currently be resolved (that is, that have bindings) in a particular context. [Ch. 2]

environment—1. In a discussion of systems, everything surrounding a system that is not viewed as part of that system. The distinction between a system and its environment is a choice based on the purpose, ease of description, and minimization of interconnections. [Ch. 1] 2. In an interpreter, the state on which the interpreter should perform the actions directed by program instructions. [Ch. 2]

environment reference—The component of an interpreter that tells the interpreter where to find its environment. [Ch. 2]

erasure—An error in a string of bits, bytes, or groups of bits in which an identified bit, byte, or group of bits is missing or has indeterminate value. [Ch. 8]

ergodic—A property of some time-dependent probabilistic processes: that the (usually easier to measure) ensemble average of some parameter measured over a set of elements subject to the process is the same as the time average of that parameter of any single element of the ensemble. [Ch. 8]

error—Informally, a label for an incorrect data value or control signal caused by an active fault. If there is a complete formal specification for the internal design of a module, an error is a violation of some assertion or invariant of the specification. An error in a module is not identical to a failure of that module, but if an error is not masked, it may lead to a failure of the module. [Ch. 8]

error containment—Limiting how far the effects of an error propagate. A module is normally designed to contain errors in such a way that the effects of an error appear in a predictable way at the module's interface. [Ch. 8]

error correction—A scheme to set to the correct value a data value or control signal that is in error. Compare with *error detection*. [Ch. 8]

error-correction code—a method of encoding stored or transmitted data with a modest amount of redundancy, in such a way that any errors during storage or transmission will, with high probability, lead to a decoding that is identical to the original data. See also the general definition of *error correction*. Compare with *error-detection code*. [Ch. 7]

error detection—A scheme to discover that a data value or control signal is in error. Compare with *error correction*. [Ch. 8]

error-detection code—a method of encoding stored or transmitted data with a small amount of redundancy, in such a way that any errors during storage or transmission will, with high probability, lead to a decoding that is obviously wrong. Compare with *error-*

correction code and *checksum*. See also the general definition of *error detection*. Compare with *error-correction code* and *checksum*. [Ch. 7]

Ethernet—A widely used broadcast network in which all participants share a common wire and can hear one another transmit. Ethernet is characterized by a transmit protocol in which a station wishing to send data first listens to ensure that no one else is sending, and then continues to monitor the network during its own transmission to see if some other station has tried to transmit at the same time, an error known as a **collision**. This protocol is named **Carrier Sense Multiple Access with Collision Detection**, abbreviated CSMA/CD. [Ch. 7]

eventcount—A special type of shared variable used for sequence coordination. It supports two primary operations: AWAIT and ADVANCE. An eventcount is a counter that is incremented atomically, using ADVANCE, while other threads wait for the counter to reach a certain value using AWAIT. Eventcounts are often used in combination with sequencers. [Ch. 5]

eventual consistency—A requirement that at some unspecified time following an update to a collection of data, if there are no more updates, the memory model for that collection will hold. [Ch. 10]

exactly-once—A protocol assurance that the intended operation or message delivery was performed both at-least-once and at-most-once. [Ch. 4]

exception—An interrupt event that pertains to the thread that a processor is currently running. [Ch. 5]

explicit context reference—For a name or an object, an associated reference to the context in which that name, or all names contained in that object, are to be resolved. Compare with *default context reference*. [Ch. 2]

explicitness—A property of a message in a security protocol: if a message is explicit, then the message contains all the information necessary for a receiver to reliably determine that the message is part of a particular run of the protocol with a specific function and set of participants. [Ch. 11]

exponential backoff—An adaptive procedure used to set a timer, for example, to wait for congestion to dissipate. Each time the timer setting proves to be too small, the action doubles (or, more generally, multiplies by a constant greater than one) the length of its next timer setting. The intent is obtain a suitable timer value as quickly as possible. See also *exponential random backoff*. [Ch. 7]

exponential random backoff—A form of *exponential backoff* in which an action that repeatedly encounters interference repeatedly doubles (or, more generally, multiplies by a constant greater than one) the size of an interval from which it randomly chooses its next delay before retrying. The intent is that by randomly changing the timing relative to other, interfering actions, the interference will not recur. [Ch. 9]

export—In naming, to provide a name for an object that other objects can use. [Ch. 2]

fail-fast—Describes a system or module design that contains detected errors by reporting

at its interface that its output may be incorrect. Compare with *fail-stop*. [Ch. 8]

fail-safe—Describes a system design that detects incorrect data values or control signals and forces them to values that, even if not correct, are known to allow the system to continue operating safely. [Ch. 8]

fail-secure—Describes an application of fail-safe design to information protection: a failure is guaranteed not to allow unauthorized access to protected information. In early work on fault tolerance, this term was also occasionally used as a synonym for *fail-fast*. [Ch. 8]

fail-soft—Describes a design in which the system specification allows errors to be masked by degrading performance or disabling some functions in a predictable manner. [Ch. 8]

fail-stop—Describes a system or module design that contains detected errors by stopping the system or module as soon as possible. Compare with *fail-fast*, which does not require other modules to take additional action, such as setting a timer, to detect the failure. [Ch. 8]

fail-vote—Describes an *N*-modular redundancy system with a majority voter. [Ch. 8]

failure—The outcome when a component or system does not produce the intended result at its interface. Compare with **fault**. [Ch. 8]

failure tolerance—A measure of the ability of a system to mask active faults and continue operating correctly. A typical measure counts the number of contained components that can fail without causing the system to fail. [Ch. 8]

fault—A defect in materials, design, or implementation that may (or may not) cause an error and lead to a failure. (Compare with *failure*.) [Ch. 8]

fault avoidance—A strategy to design and implement a component with a probability of faults that is so low that it can be neglected. When applied to software, fault avoidance is sometimes called *valid construction*. [Ch. 8]

fault tolerance—A set of techniques that involve noticing active faults and lower-level subsystem failures and masking them, rather than allowing the resulting errors to propagate. [Ch. 8]

file—A popular memory abstraction to durably store and retrieve data. A typical interface for a file consists of procedures to OPEN the file, to READ and WRITE regions of the file, and to CLOSE the file. [Ch. 2]

fingerprint—Another term for a *witness*. [Ch. 10]

first-come, first-served (FCFS) scheduling policy—A scheduling policy in which requests are processed in the order in which they arrive. [Ch. 6]

first-in, first-out (FIFO) policy—A particular page-removal policy for a multilevel memory system. FIFO chooses to remove the page that has been in the primary device the longest. [Ch. 6]

flow control—1. In networks, an end-to-end protocol between a fast sender and a slow recipient, a mechanism that limits the sender's data rate so that the recipient does not receive data faster than it can handle. [Ch. 7] 2. In security, a system that allows

untrusted programs to work with sensitive data but confines all program outputs to prevent unauthorized disclosure. [Ch. 11]

force—(v.) When output may be buffered, to ensure that a previous output value has actually been written to durable storage or sent as a message. Caches that are not write-through usually have a feature that allows the invoker to force some or all of their contents to the secondary storage medium. [Ch. 9]

forward error correction—A technique for controlling errors in which enough redundancy to correct anticipated errors is applied before an error occurs. Forward error correction is particularly applicable when the original source of the data value or control signal will not be available to recalculate or resend it. Compare with *backward error correction*. [Ch. 8]

forward secrecy—A property of a security protocol. A protocol has forward secrecy if information, such as an encryption key, deduced from a previous transcript doesn't allow an adversary to decrypt future messages. [Ch. 11]

forwarding table—A table that tells the network layer which link to use to forward a packet, based on its destination address. [Ch. 7]

fragment—1. (v.) In network protocols, to divide the payload of a packet so that it can fit into smaller packets for carriage across a link with a small maximum transmission unit. 2. (n.) The resulting pieces of payload. [Ch. 7]

frame—1. (n.) The unit of transmission in the link layer. Compare with *packet, segment,* and *message*. 2. (v.) To delimit the beginning and end of a bit, byte, frame (n.), packet, segment, or message within a stream. [Ch. 7]

freshness—A property of a message in a security protocol: if the message is fresh, it is assured not to be a replay. [Ch. 11]

full-duplex—Describes a duplex link or connection between two stations that can be used in both directions at the same time. Compare with *simplex*, *duplex*, and *half-duplex*. [Ch. 7]

gate—A predefined protected entry point into a domain. [Ch. 5]

generated name—A name created algorithmically, rather than chosen by a person. [Ch. 3]

global name—In a layered naming scheme, a name that is bound only in the outermost context layer, and thus has the same meaning to all users. [Ch. 2]

half-duplex—Describes a duplex link or connection between two stations that can be used in only one direction at a time. Compare with *simplex*, *duplex*, and *full-duplex*. [Ch. 7]

Hamming distance—in an encoding system, the number of bits in an element of a code that would have to change to transform it into a different element of the code. The Hamming distance of a code is the minimum Hamming distance between any pair of elements of the code. [Ch. 8]

hard real-time scheduling policy—A real-time scheduler in which missing a deadline may result in a disaster. [Ch. 6]

hash function—A function that algorithmically derives a relatively short, fixed-length string of bits from an arbitrarily-large block of data. The resulting short string is known as a **hash**. See also *cryptographic hash function*. [Ch. 3]

header—Information that a protocol layer adds to the front of a packet. [Ch. 7]

hierarchical routing—A routing system that takes advantage of hierarchically assigned network destination addresses to reduce the size of its routing tables. [Ch. 7]

hierarchy—A technique of organizing systems that contain many components: group small numbers of components into self-contained and stable subsystems that then become components of larger self-contained and stable subsystems, and so on. [Ch. 1]

hit ratio—In a multilevel memory, the fraction of references satisfied by the primary memory device. [Ch. 6]

hop limit—A network-layer protocol field that acts as a safety net to prevent packets from endlessly circulating in a network that has inconsistent forwarding tables. [Ch. 7]

hot swap—To replace modules in a system while the system continues to provide service. [Ch. 8]

idempotent—Describes an action that can be interrupted and restarted from the beginning any number of times and still produce the same result as if the action had run to completion without interruption. The essential feature of an idempotent action is that if there is any question about whether or not it completed, it is safe to do it again. "Idempotent" is correctly pronounced with the accent on the second syllable, not on the first and third. [Ch. 4]

identifier—A synonym for *name*, sometimes used to avoid an implication that the name might be meaningful to a person rather than to a machine. [Ch. 3]

illegal instruction—An instruction that an interpreter is not equipped to execute because it is not in the interpreter's instruction repertoire or it has an out-of-range operand (for example, an attempt to divide by zero). An illegal instruction typically causes an interrupt. [Ch. 2]

incommensurate scaling—A property of most systems, that as the system grows (or shrinks) in size, not all parts grow (or shrink) at the same rate, thus stressing the system design. [Ch. 1]

incremental backup—A backup copy that contains only data that has changed since making the previous backup copy. [Ch. 10]

indirect name—A name that is bound to another name in the same name space. "Symbolic link", "soft link", and "shortcut" are other words used for this concept. Some operating systems also define the term *alias* to have this meaning rather than its more general meaning of synonym. [Ch. 2]

indirection—Decoupling a connection from one object to another by interposing a name with the goal of delaying the choice of (or allowing a later change about) which object the name refers to. Indirection makes it possible to delay the choice of or change which

object is used without the need to change the object that uses it. Using a name is sometimes described as "inserting a level of indirection". [Ch. 1]

install—In a system that uses logs to achieve all-or-nothing atomicity, to write data to cell storage. [Ch. 9]

instruction reference—A characteristic component of an interpreter: the place from which it will take its next instruction. [Ch. 2]

intended load—The amount of a shared resource that a set of users would attempt to utilize if the resource had unlimited capacity. In systems that have no provision for congestion control, the intended load is equal to the offered load. The goal of congestion control is to make the offered load smaller than the intended load. Compare with *offered load*. [Ch. 7]

interleaving—A technique to improve performance by distributing apparently sequential requests to several instances of a device, so that the requests may actually be processed concurrently. [Ch. 6]

intermittent fault—A persistent fault that is active only occasionally. Compare with *transient fault*. [Ch. 8]

International Organization for Standardization (ISO)—An international non-governmental body that sets many technical and manufacturing standards including the (frequently ignored) Open Systems Interconnect (OSI) reference model for data communication networks. The short name ISO is not an acronym, it is the Greek word for "equal", chosen to be the same in all languages and always spelled in all capital letters. [Ch. 7]

interpreter—The abstraction that models the active mechanism performing computations. An interpreter comprises three components: an instruction reference, a context reference, and an instruction repertoire. [Ch. 2]

interrupt—An event that causes an interpreter to transfer control to the first instruction of a different procedure, an interrupt handler, instead of executing the next instruction. [Ch. 2]

invalidate—In a cache, to mark "do not use" or completely remove a cache entry because some event has occurred that may make the value associated with that entry incorrect. [Ch. 10]

isochronous (From Greek roots meaning "equal" and "time")—Describes a communication link over which data is sent in frames whose length is fixed in advance and whose timing relative to other frames is precisely predictable. Compare with *asynchronous*. [Ch. 7]

jitter—In real-time applications, variability in the delivery times of successive data elements. [Ch. 7]

job—The unit of granularity on which threads are scheduled. A job corresponds to the burst of activity of a thread between two idle periods. [Ch. 6]

journal storage—Storage in which a WRITE or PUT appends a new value, rather than overwriting a previously stored value. Compare with *cell storage*. [Ch. 9]

kernel—A trusted intermediary that virtualizes resources for mutually distrustful modules running on the same computer. Kernel modules typically run with kernel mode enabled. [Ch. 5]

kernel mode—A feature of a processor that when set allows threads to use special processor features (e.g., the page-map address register) that are disallowed to threads that run with kernel mode disabled. Compare with *user mode*. [Ch. 5]

key-based cryptographic transformation—A cryptographic transformation for which successfully meeting the cryptographic goals depends on the secrecy of some component of the transformation. That component is called a cryptographic key, and a usual design is to make that key a small, modular, separable, and easily changeable component. [Ch. 11]

key distribution center (KDC)—A principal that authenticates other principals to one another and also provides one or more temporary cryptographic keys for communication between other principals. [Ch. 11]

latency—The delay between a change at the input to a system and the corresponding change at its output. [Ch. 2] As used in reliability, the time between when a fault becomes active and when the module in which the fault occurred either fails or detects the resulting error. [Ch. 8]

latent fault—A fault that is not currently causing an error. Compare with *active fault*. [Ch. 8]

layering—A technique of organizing systems in which the designer builds on an interface that is already complete (a lower layer), to create a different complete interface (an upper layer). [Ch. 1]

least-recently-used (LRU) policy—A popular page-removal policy for a multilevel memory system. LRU chooses to remove the page that has not been used the longest. [Ch. 6]

lexical scope—Another term for *static scope*. [Ch. 2]

limited name space—A name space in which a limited number of names can be expressed and therefore names must be allocated, deallocated, and reused. [Ch. 3]

link—1 (n.) Another term for a *synonym* (usually called a hard link) or an indirect name (usually called a soft or symbolic link). 2 (v.) Another term for *bind*. [Ch. 2] 3. (n.) In data communication, a communication path between two points. [Ch. 7]

link layer—The communication system layer that moves data directly from one physical point to another. [Ch. 7]

list system—A design for an access control mechanism in which each protected object is associated with a list of authorized principals. [Ch. 11]

livelock—An undesirable interaction among a group of threads in which each thread

begins a sequence of actions, discovers that it cannot complete the sequence because actions of other threads have interfered, and begins again, endlessly. [Ch. 5]

locality of reference—A property of most programs that memory references tend to be clustered in both time and address space. [Ch. 6]

lock—A flag associated with a data object, set by a thread to warn concurrent threads that the object is in use and that it may be a mistake for other threads to read or write it. Locks are one technique used to achieve before-or-after atomicity. [Ch. 5]

lock point—In a system that provides before-or-after atomicity by locking, the first instant in a before-or-after action when every lock that will ever be in its lock set has been acquired. [Ch. 9]

lock set—The collection of all locks acquired during the execution of a before-or-after action. [Ch. 9]

lock-step protocol—In networking, any transport protocol that requires acknowledgment of the previously sent message, segment, packet, or frame before sending another message, segment, packet, or frame to the same destination. Sometimes called a *stop and wait* protocol. Compare with *pipeline*. [Ch. 7]

log—1. (n.) A specialized use of journal storage to maintain an append-only record of some application activity. Logs are used to implement all-or-nothing actions, for performance enhancement, for archiving, and for reconciliation. 2. (v.) To append a record to a log. [Ch. 9]

logical copy—A replica that is organized in a form determined by a higher layer. An example is a replica of a file system that is made by copying one file at a time. Analogous to logical locking. Compare with physical copy. [Ch. 10]

logical locking—Locking of higher-layer data objects such as records or fields of a database. Compare with *physical locking*. [Ch. 9]

Manchester code—A particular type of phase encoding in which each bit is represented by two bits of opposite value. [Ch. 7]

margin—The amount by which a specification is better than necessary for correct operation. The purpose of designing with margins is to mask some errors. [Ch. 8]

mark point—1. (adj.) An atomicity-assuring discipline in which each newly created action n must wait to begin reading shared data objects until action $(n - 1)$ has marked all of the variables it intends to modify. 2. (n.) The instant at which an action has marked all of the variables it intends to modify. [Ch. 9]

marshal/unmarshal—To marshal is to transform the internal representation of one or more pieces of data into a form that is more suitable for transmission or storage. The opposite action, to unmarshal, is to parse marshaled data into its constituent data pieces and transform those pieces into a suitable internal representation. [Ch. 4]

maskable error—An error or class of errors that is detectable and for which a systematic recovery strategy can in principle be devised. Compare with *detectable error* and *tolerated*

error. [Ch. 8]

masking—As used in reliability, containing an error within a module in such a way that the module meets its specifications as if the error had not occurred. [Ch. 8]

master—In a multiple-site replication scheme, the site to which updates are directed. Compare with *slave*. [Ch. 10]

maximum transmission unit (MTU)—A limit on the size of a packet, imposed to control the time commitment involved in transmitting the packet, to control the amount of loss if congestion causes the packet to be discarded, and to keep low the probability of a transmission error. [Ch. 7]

mean time between failures (MTBF)—The sum of MTTF and MTTR for the same component or system. [Ch. 8]

mean time to failure (MTTF)—The expected time that a component or system will operate continuously without failing. "Time" is sometimes measured in cycles of operation. [Ch. 8]

mean time to repair (MTTR)—The expected time to replace or repair a component or system that has failed. The term is sometimes written as "mean time to restore service", but it is still abbreviated MTTR. [Ch. 8]

mediation—Before a service performs a requested operation, determining which principal is associated with the request and whether the principal is authorized to request the operation. [Ch. 11]

memory—The abstraction for remembering data values, using READ and WRITE operations. The WRITE operation specifies a value to be remembered and a name by which that value can be recalled in the future. See also *storage*. [Ch. 2]

memoryless—A property of some time-dependent probabilistic processes, that the probability of what happens next does not depend on what has happened before. [Ch. 8]

memory manager—A device located between a processor and memory that translates virtual to physical addresses and checks that memory references by the thread running on the processor are in the thread's domain(s). [Ch. 5]

memory-mapped I/O—An interface that allows an interpreter to communicate with an I/O module using LOAD and STORE instructions that have ordinary memory addresses. [Ch. 2]

message—The unit of communication at the application level. The length of a message is determined by the application that sends it. Since a network may have a maximum size for its unit of transmission, the end-to-end layer divides a message into one or more segments, each of which is carried in a separate packet. Compare with *frame* (n.), *segment*, and *packet*. [Ch. 7]

message authentication—The verification of the integrity of the origin and the data of a message. [Ch. 11]

message authentication code (MAC)—An authentication tag computed with shared-secret cryptography. MAC is sometimes used as a verb in security jargon, as in "Just to be careful, let's MAC the address field of that message." [Ch. 11]

metadata—Information about an object that is not part of the object itself. Examples are the name of the object, the identity of its owner, the date it was last modified, and the location in which it is stored. [Ch. 3]

microkernel—A kernel organization in which most operating system comonents run in separate, user-mode address spaces. [Ch. 5]

mirror—(n.) One of a set of replicas that is created or updated synchronously. Compare with *primary copy* and *backup copy*. Sometimes used as a verb, as in "Let's mirror that data by making three replicas." [Ch. 8]

missing-page exception—The event when an addressed page is not present in the primary device and the virtual memory manager has to move the page in from a secondary device. The literature also uses the term *page fault*. [Ch. 6]

modular sharing—Sharing of an object without the need to know details of the implementation of the shared object. With respect to naming, modular sharing is sharing without the need to know the names that the shared object uses to refer to its components. [Ch. 3]

module—A system component that can be separately designed, implemented, managed, and replaced. [Ch. 1]

monolithic kernel—A kernel organization in which most operating system components run in a single, kernel-mode address space. [Ch. 5]

most-recently-used (MRU) policy—A page-removal policy for a multilevel memory system. MRU chooses for removal the most recently used page in the primary device. [Ch. 6]

MTU discovery—A procedure that systematically discovers the smallest maximum transmission unit along the path between two network attachment points. [Ch. 7]

multihomed—Describes a single physical interface between the network layer and the end-to-end layer that is associated with more than one network attachment point, each with its own network-layer address. [Ch. 7]

multilevel memory—Memory built out of two or more different memory devices that have significantly different latencies and cost per bit. [Ch. 6]

multiple lookup—A name-mapping algorithm that tries several contexts in sequence, looking for the first one that can successfully resolve a presented name. [Ch. 2]

multiplexing—Sharing a communication link among several, usually independent, simultaneous communications. The term is also used in layered protocol design when several different higher-layer protocols share the same lower-layer protocol. [Ch. 7]

multipoint—Describes communication that involves more than two parties. A multipoint link is a single physical medium that connects several parties. A multipoint protocol

coordinates the activities of three or more participants. [Ch. 7]

N + 1 redundancy—When a load can be handled by sharing it among *N* equivalent modules, the technique of installing *N* + 1 or more of the modules, so that if one fails the remaining modules can continue to handle the full load while the one that failed is being repaired. [Ch. 8]

N-modular redundancy (NMR)—A redundancy technique that involves supplying identical inputs to *N* equivalent modules and connecting the outputs to one or more voters. [Ch. 8]

N-version programming—The software version of *N*-modular redundancy. *N* different teams each independently write a program from its specifications. The programs then run in parallel, and voters compare their outputs. [Ch. 8]

nam*e*—A designator or an identifier of an object or value. A name is an element of a name space. [Ch. 2]

name conflict—An occurrence when, for some reason, it seems necessary to bind the same name to two different values at the same time in the same context. Usually, a result of encountering a preexisting name in a naming scheme that does not provide modular sharing. When names are algorithmically generated, name conflicts are called *collisions*. [Ch. 3]

name-mapping algorithm—See *naming scheme.* [Ch. 2]

name space—The set of all possible names of a particular naming scheme. A name space is defined by a set of symbols from some alphabet together with a set of syntax rules that define which names are members of the name space. [Ch. 2]

name-to-key binding—A binding between a principal identifier and a cryptographic key. [Ch. 11]

naming hierarchy—A naming network that is constrained to a tree-structured form. The root used for interpretation of absolute path names (which in a naming hierarchy are sometimes called "tree names") is normally the base of the tree. [Ch. 2]

naming network—A naming scheme in which contexts are named objects and any context may contain a binding for any other context, as well as for any non-context object. An object in a naming network is identified by a multicomponent path name that traces a path through the naming network from some starting point, which may be either a default context or a root. [Ch. 2]

naming scheme—A particular combination of a name space, a universe of values (which may include physical objects) that can be named, and a name-mapping algorithm that provides a partial mapping from the name space to the universe of values. [Ch. 2]

negative acknowledgment (NAK or NACK)—A status report from a recipient to a sender asserting that some previous communication was not received or was received incorrectly. The usual reason for sending a negative acknowledgment is to avoid the delay that would be incurred by waiting for a timer to expire. Compare with *acknowledgment.* [Ch. 7]

network—A communication system that interconnects more than two things. [Ch. 7]

network address—In a network, the identifier of the source or destination of a packet. [Ch. 7]

network attachment point—The place at which the network layer accepts or delivers payload data to and from the end-to-end layer. Each network attachment point has an identifier, its *address*, that is unique within that network. A network attachment point is sometimes called an *access point*, and in ISO terminology, a *Network Services Access Point* (NSAP). [Ch. 7]

network layer—The communication system layer that forwards data through intermediate links to carry it to its intended destination. [Ch. 7]

non-discretionary access control—A property of an access control system. In a non-discretionary access control system, some principal other than the owner has the authority to decide which principals have to access the object. Compare with *discretionary access control*. [Ch. 11]

non-preemptive scheduling—A scheduling policy in which threads run until they explicitly yield or wait. [Ch. 5]

non-volatile memory—A kind of memory that does not require a continuous source of power, so it retains its content when its power supply is off. The phrase "stable storage" is a common synonym. Compare with *volatile memory*. [Ch. 2]

nonce—A unique identifier that should never be reused. [Ch. 7]

object—As used in naming, any software or hardware structure that can have a distinct name. [Ch. 2]

offered load—The amount of a shared service that a set of users attempt to utilize. *Presented load* is an occasionally encountered synonym. [Ch. 6]

opaque name—In a modular system, a name that, from the point of view of the current module, carries no overloading that the module knows how to interpret. [Ch. 3]

operating system—A collection of programs that provide services such as abstraction and management of hardware devices and features such as libraries of commonly needed procedures, all of which are intended to make it easier to write application programs. [Ch. 2]

optimal (OPT) page-removal policy—An unrealizable page-removal policy for a multilevel memory system. The optimal policy removes from primary memory the page that will not be used for the longest time. Because identifying that page requires knowing the future, the optimal policy is not implementable in practice. Its utility is that after any particular reference string has been observed, one can then simulate the operation of that reference string with the optimal policy, to compare the number of missing-page exceptions with the number obtained when using other, realizable policies. [Ch. 6]

optimistic concurrency control—A concurrency control scheme that allows concurrent threads to proceed even though a risk exists that they will interfere with each other, with

the plan of detecting whether there actually is interference and, if necessary, forcing one of the threads to abort and retry. Optimistic concurrency control is an effective technique in situations where interference is possible but not likely. Compare with *pessimistic concurrency control.* [Ch. 9]

origin authenticity—Authenticity of the claimed origin of a message. Compare with *data integrity.* [Ch. 11]

overload—When offered load exceeds the capacity of a service for a specified period of time. [Ch. 6]

overloaded name—A name that does more than simply identify an object; it also carries other information, such as the type of the object, the date it was modified, or how to locate it. Overloading is commonly encountered when a system has not made suitable provision to handle metadata. Contrast with *pure name.* [Ch. 3]

packet—The unit of transmission of the network layer. A packet consists of a segment of payload data, accompanied by guidance information that allows the network to forward it to the network attachment point that is intended to receive the data carried in the packet. Compare with *frame* (n.), *segment,* and *message.* [Ch. 7]

packet forwarding—In the network layer, upon receiving a packet that is not destined for the local end layer, to send it out again along some link with the intention of moving the packet closer to its destination. [Ch. 7]

packet switch—A specialized computer that forwards packets in a data communication network. Sometimes called a *packet forwarder* or, if it also implements an adaptive routing algorithm, a *router.* [Ch. 7]

page—In a page-based virtual memory system, the unit of translation between virtual addresses and physical addresses. [Ch. 5]

page fault—See *missing-page exception.*

page map—Data structure employed by the virtual memory manager to map virtual addresses to physical addresses. [Ch. 5]

page-map address register—A processor register maintained by the thread manager. It contains a pointer to the page map used by the currently active thread, and it can be changed only when the processor is in kernel mode. [Ch. 5]

page-removal policy—A policy for deciding which page to move from the primary to the secondary device to make a space to bring in a new page. [Ch. 6]

page table—A particular form of a page map, in which the map is organized as an array indexed by page number. [Ch. 5]

pair-and-compare—A method for constructing fail-fast modules from modules that do not have that property, by connecting the inputs of two replicas of the module together and connecting their outputs to a comparator. When one repairs a failed pair-and-compare module by replacing the entire two-replica module with a spare, rather than identifying and replacing the replica that failed, the method is called **pair-and-spare.**

[Ch. 8]

pair-and-spare—See *pair-and-compare*.

parallel transmission—A scheme for increasing the data rate between two modules by sending data over several parallel lines that are coordinated by the same clock. [Ch. 7]

partition—To divide a job up and assign it to different physical devices, with the intent that a failure of one device does not prevent the entire job from being done. [Ch. 8]

password—A secret character string used to authenticate the claimed identity of an individual. [Ch. 11]

path name—A name with internal structure that traces a path through a naming network. Any prefix of a path name can be thought of as the explicit context reference to use for resolution of the remainder of the path name. See also *absolute path name* and *relative path name*. [Ch. 2]

path selection—In a network-layer routing protocol, when a participant updates its own routing information with new information learned from an exchange with its neighbors. [Ch. 7]

payload—In a layered description of a communication system, the data that a higher layer has asked a lower layer to send; used to distinguish that data from the headers and trailers that the lower layer adds. (This term seems to have been borrowed from the transportation industry, where it is used frequently in aerospace applications.) [Ch. 7]

pending—A state of an all-or-nothing action, when that action has not yet either committed or aborted. Also used to describe the value of a variable that was set or changed by a still-pending all-or-nothing action. [Ch. 9]

persistence—A property of an active agent such as an interpreter that, when it detects it has failed, it keeps trying until it succeeds. Compare with *stability* and *durability,* terms that have different technical definitions as explained in Sidebar 2.7. The adjective "persistent" is used in some contexts as a synonym for stable and sometimes also in the sense of immutable. [Ch. 2]

persistent fault—A fault that cannot be masked by retry. Compare with *transient fault* and *intermittent fault*. [Ch. 8]

persistent sender—A transport protocol participant that, by sending the same message repeatedly, tries to ensure that at least one copy of the message gets delivered. [Ch. 7]

pessimistic concurrency control—A concurrency control scheme that forces a thread to wait if there is any chance that by proceeding it may interfere with another, concurrent, thread. Pessimistic concurrency control is an effective technique in situations where interference between concurrent threads has a high probability. Compare with *optimistic concurrency control*. [Ch. 9]

phase encoding—A method of encoding data for digital transmission in which at least one level transition is associated with each transmitted bit, to simplify framing and recovery of the sender's clock. [Ch. 7]

physical address—An address that is translated geometrically to read or write data stored on a device. Compare with *virtual address*. [Ch. 5]

physical copy—A replica that is organized in a form determined by a lower layer. An example is a replica of a disk that is made by copying it sector by sector. Analogous to *physical locking*. Compare with *logical copy*. [Ch. 10]

physical locking—Locking of lower-layer data objects, typically chunks of data whose extent is determined by the physical layout of a storage medium. Examples of such chunks are disk sectors or even an entire disk. Compare with *logical locking*. [Ch. 9]

piggybacking—In an end-to-end protocol, a technique for reducing the number of packets sent back and forth by including acknowledgments and other protocol state information in the header of the next packet that goes to the other end. [Ch. 7]

pipeline—In networking, a transport protocol design that allows sending a packet before receiving an acknowledgment of the packet previously sent to the same destination. Contrast with *lock-step protocol*. [Ch. 7]

plaintext—The result of decryption. Also sometimes used to describe data that has not been encrypted, as in "The mistake was sending that message as plaintext." Compare with *ciphertext*. [Ch. 11]

point-to-point—Describes a communication link between two stations, as contrasted with a broadcast or multipoint link. [Ch. 7]

polling—A style of interaction between threads or between a processor and a device in which one periodically checks whether the other needs attention. [Ch. 5]

port—In an end-to-end transport protocol, the multiplexing identifier that tells which of several end-to-end applications or application instances should receive the payload. [Ch. 7]

preemptive scheduling—A scheduling policy in which a thread manager can interrupt and reschedule a running thread at any time. [Ch. 5]

prepaging—An optimization for a multilevel memory manager in which the manager predicts which pages might be needed and brings them into the primary memory before the application demands them. Compare with *demand algorithm*. [Ch. 6]

prepared—In a layered or multiple-site all-or-nothing action, a state of a component action that has announced that it can, on command, either commit or abort. Having reached this state, it awaits a decision from the higher-layer coordinator of the action. [Ch. 9]

presentation protocol—A protocol that translates semantics and data of the network to match those of the local programming environment. [Ch. 7]

presented load—See *offered load*.

preventive maintenance—Active intervention intended to increase the mean time to failure of a module or system and thus improve its reliability and availability. [Ch. 8]

primary copy—Of a set of replicas that are not written or updated synchronously, the one that is considered authoritative and, usually, written or updated first. (Compare with

mirror and *backup copy*.) [Ch. 10]

primary device—In a multilevel memory system, the memory device that is faster and usually more expensive and thus smaller. Compare with *secondary device*. [Ch. 6]

principal—The representation inside a computer system of an agent (a person, a computer, a thread) that makes requests to the security system. A principal is the entity in a computer system to which authorizations are granted; thus, it is the unit of accountability and responsibility in a computer system. [Ch. 11]

priority scheduling policy—A scheduling policy in which some jobs have priority over other jobs. [Ch. 6]

privacy—A socially defined ability of an individual (or organization) to determine if, when, and to whom personal (or organizational) information is to be released and also what limitations should apply to use of released information. [Ch. 11]

private key—In public-key cryptography, the cryptographic key that must be kept secret. Compare with *public key*. [Ch. 11]

processing delay—In a communication network, that component of the overall delay contributed by computation that takes place in various protocol layers. [Ch. 7]

program counter—A processor register that holds the reference to the current or next instruction that the processor is to execute. [Ch. 2]

progress—A desirable guarantee provided by an atomicity-assuring mechanism: that despite potential interference from concurrency some useful work will be done. An example of such a guarantee is that the atomicity-assuring mechanism will not abort at least one member of the set of concurrent actions. In practice, lack of a progress guarantee can sometimes be repaired by using exponential random backoff. In formal analysis of systems, progress is one component of a property known as "liveness". Progress is an assurance that the system will move toward some specified goal, whereas liveness is an assurance that the system will eventually reach that goal. [Ch. 9]

propagation delay—In a communication network, the component of overall delay contributed by the velocity of propagation of the physical medium used for communication. [Ch. 7]

propagation of effects—A property of most systems: a change in one part of the system causes effects in areas of the system that are far removed from the changed part. A good system design tends to minimize propagation of effects. [Ch. 1]

protection—1. Synonym for *security*. 2. Sometimes used in a narrower sense to denote mechanisms and techniques that control the access of executing programs to information. [Ch. 11]

protection group—A principal that is shared by more than one user. [Ch. 11]

protocol—An agreement between two communicating parties, for example on the messasges and format of data that they intend to exchange. [Ch. 7]

public key—In public-key cryptography, the key that can be published (i.e., the one that

doesn't have to be kept secret). Compare with *private key*. [Ch. 11]

public-key cryptography—A key-based cryptographic transformation that can provide both confidentiality and authenticity of messages without the need to share a secret between sender and recipient. Public-key systems use two cryptographic keys, one of which must be kept secret but does not need to be shared. [Ch. 11]

publish/subscribe—A communication style using a trusted intermediary. Clients push or pull messages to or from an intermediary. The intermediary determines who actually receives a message and if a message should be fanned out to multiple recipients. [Ch. 4]

pure name—A name that is not overloaded in any way. The only operations that apply to a pure name are COMPARE, RESOLVE, BIND, and UNBIND. Contrast with *overloaded name*. [Ch. 3]

purging—A technique used in some N-modular redundancy designs, in which the voter ignores the output of any replica that, at some time in the past, disagreed with several others. [Ch. 8]

qualified name—A name that includes an explicit context reference. [Ch. 2]

quench—(n.) An administrative message sent by a packet forwarder to another forwarder or to an end-to-end-layer sender asking that the forwarder or sender stop sending data or reduce its rate of sending data. [Ch. 7]

queuing delay—In a communication network, the component of overall delay that is caused by waiting for a resource such as a link to become available. [Ch. 7]

quorum—A partial set of replicas intended to improve availability. One defines a read quorum and a write quorum that intersect, with the goal that for correctness it is sufficient to read from a read quorum and write to a write quorum. [Ch. 10]

race condition—A timing-dependent error in thread coordination that may result in threads computing incorrect results (for example, multiple threads simultaneously try to update a shared variable that they should have updated one at a time). [Ch. 5]

RAID—An acronym for Redundant Array of Independent (or Inexpensive) Disks, a set of techniques that use a controller and multiple disk drives configured to improve some combination of storage performance or durability. A RAID system usually has an interface that is electrically and programmatically identical to a single disk, thus allowing it to transparently replace a single disk. [Ch. 2]

random access memory—A memory device for which the latency for memory cells chosen at random is approximately the same as the latency obtained by choosing cells in the pattern best suited for that memory device. [Ch. 2]

random drop—A strategy for managing an overloaded resource: the system refuses service to a queue member chosen at random. [Ch. 7]

random early detection *(RED)*—A combination of random drop and early drop. [Ch. 7]

rate monotonic scheduling policy—A policy that schedules periodic jobs for a real-time system. Each job receives in advance a priority that is proportional to the frequency of

the occurrence of that job. The scheduler always runs the highest priority job, preempting a running job, if necessary. [Ch. 6]

Read and Set Memory (*RSM*)—A hardware or software function used primarily for implementing locks. RSM loads a value from a memory location into a register and stores another value in the same memory location. The important property of RSM is that no other loads and stores by concurrent threads can come between the load and the store of an RSM. RSM is nearly always implemented as a hardware instruction. [Ch. 5]

read/write coherence—A property of a memory, that a READ always returns the result of the most recent WRITE. [Ch. 2]

ready/acknowledge protocol—A data transmission protocol in which each transmission is framed by a ready signal from the sender and an acknowledge signal from the receiver. [Ch. 7]

real time—1. (adj.) Describes a system that requires delivery of results before some deadline. 2. (n.) The wall-clock sequence that an all-seeing observer would associate with a series of actions. [Ch. 6]

real-time scheduling policy—A scheduler that attempts to schedule jobs in such a way that all jobs complete before their deadlines. [Ch. 6]

reassembly—Reconstructing a message by arranging, in correct order, the segments it was divided into for transmission. [Ch. 7]

reconciliation—A procedure that compares replicas that are intended to be identical and repairs any differences. [Ch. 10]

recursive name resolution—A method of resolving path names. The least significant component of the path name is looked up in the context named by the remainder of the path name, which must thus be resolved first. [Ch. 2]

redo action—An application-specified action that, when executed during failure recovery, produces the effect of some committed component action whose effect may have been lost in the failure. (Some systems call this a "do action". Compare with *undo action*.) [Ch. 9]

redundancy—Extra information added to detect or correct errors in data or control signals. [Ch. 8]

reference—(n.) Use of a name by an object to refer to another object. In grammatical English, the corresponding verb is "to refer to". In computer jargon, the non-standard verb "to reference" appears frequently, and the coined verb "dereference" is a synonym for *resolve*. [Ch. 2]

reference string—The string of addresses issued by a thread during its execution (typically the string of the virtual addresses issued by a thread's execution of LOAD and STORE instructions; it may also include the addresses of the instructions themselves). [Ch. 6]

relative path name—A path name that the name resolver resolves in a default context provided by the environment. [Ch. 2]

reliability—A statistical measure, the probability that a system is still operating at time t, given that it was operating at some earlier time t_0. [Ch. 8]

reliable delivery—A transport protocol assurance: it provides both at-least-once delivery and data integrity. [Ch. 7]

remote procedure call (RPC)—A stylized form of client/service interaction in which each request is followed by a response. Usually, remote procedure call systems also provide marshaling and unmarshaling of the request and the response data. The word "procedure" in "remote procedure call" is misleading, since RPC semantics are different from those of an ordinary procedure call: for example, RPC specifically allows for clients and the service to fail independently. [Ch. 4]

repair—An active intervention to fix or replace a module that has been identified as failing, preferably before the system of which it is a part fails. [Ch. 8]

repertoire—The set of operations or actions an interpreter is prepared to perform. The repertoire of a general-purpose processor is its instruction set. [Ch. 2]

replica—1. One of several identical modules that, when presented with the same inputs, is expected to produce the same output. 2. One of several identical copies of a set of data. [Ch. 8]

replicated state machine—A method of performing an update to a set of replicas that involves sending the update request to each replica and performing it independently at each replica. [Ch. 10]

replication—The technique of using multiple replicas to achieve fault tolerance. [Ch. 8]

repudiate—To disown an apparently authenticated message. [Ch. 11]

request—The message sent from a client to a service. [Ch. 4]

resolve—To perform a name-mapping algorithm from a name to the corresponding value. [Ch. 2]

response—The message sent from a service to a client in response to a previous request. [Ch. 4]

roll-forward recovery—A write-ahead log protocol with the additional requirement that the application log its outcome record *before* it performs any install actions. If there is a failure before the all-or-nothing action passes its commit point, the recovery procedure does not need to undo anything; if there is a failure after commit, the recovery procedure can use the log record to ensure that cell storage installs are not lost. Also known as *redo logging*. Compare with *rollback recovery*. [Ch. 9]

rollback recovery—A write-ahead log protocol with the additional requirement that the application perform all install actions *before* logging an outcome record. If there is a failure before the all-or-nothing action commits, a recovery procedure can use the log record to undo the partially completed all-or-nothing action. Also known as *undo logging*. Compare with *roll-forward recovery*. [Ch. 9]

root—The context used for the interpretation of absolute path names. The name for the

root is usually bound to a constant value (typically, a well-known name of a lower layer) and that binding is normally built in to the name resolver at design time. [Ch. 2]

round-robin scheduling—A preemptive scheduling policy in which a thread runs for some maximum time before the next one is scheduled. When all threads have run, the scheduler starts again with the first thread. [Ch. 6]

round-trip time—In a network, the time between sending a packet and receiving the corresponding response or acknowledgment. Round-trip time comprises two (possibly different) network transit times and the time required for the correspondent to process the packet and prepare a response. [Ch. 7]

router—A packet forwarder that also participates in a routing algorithm. [Ch. 7]

routing algorithm—An algorithm intended to construct consistent, efficient forwarding tables. A routing algorithm can be either *centralized*, which means that one node calculates the forwarding tables for the entire network, or *decentralized*, which means that many participants perform the algorithm concurrently. [Ch. 7]

scheduler—The part of the thread manager that implements the policy for deciding which thread to run. Policies can be preemptive or non-preemptive. [Ch. 5]

scope—In a layered naming scheme, the set of contexts in which a particular name is bound to the same value. [Ch. 2]

search—As used in naming, a synonym for *multiple lookup*. This usage of the term is a highly constrained form of the more general definition of search as used in information retrieval and full-text search systems: to locate all instances of records that match a given query. [Ch. 2]

search path—A default context reference that consists of the identifiers of the contexts to be used in a multiple lookup name resolution. The word "path" as used here has no connection with its use in *path name,* and the word "search" has only a distant connection with the concept of key word search. [Ch. 2]

secondary device—In a multilevel memory system, the memory device that is larger but also usually slower. Compare with *primary device*. [Ch. 6]

secrecy—Synonym for *confidentiality*. [Ch. 11]

secure area—A physical space or a virtual address space in which confidential information can be safely confined. [Ch. 11]

secure channel—A communication channel that can safely send information from one secure area to another. The channel may provide confidentiality or authenticity or, more commonly, both. [Ch. 11]

security—The protection of information and information systems against unauthorized access or modification of information, whether in storage, processing, or transit, and against denial of service to authorized users. [Ch. 11]

security protocol—A message protocol designed to achieve some security objective (e.g., authenticating a sender). Designers of security protocols must assume that some of the

communicating parties are adversaries. [Ch. 11]

segment—1. A numbered block of contiguously addressed virtual memory, the block having a range of memory addresses starting with address zero and ending at some specified size. Programs written for a segment-based virtual memory issue addresses that are really two numbers: the first identifies the segment number, and the second identifies the address within that segment. The memory manager must translate the segment number to determine where in real memory the segment is located. The second address may also require translation using a page map. [Ch. 5] 2. In a communication network, the data that the end-to-end layer gives to the network layer for forwarding across the network. A segment is the payload of a packet. Compare with *frame* (n.), *message*, and *packet*. [Ch. 7]

self-pacing—A property of some transmission protocols. A self-pacing protocol automatically adjusts its transmission rate to match the bottleneck data rate of the network over which it is operating. [Ch. 7]

semaphore—A special type of shared variable for sequence coordination among several concurrent threads. A semaphore supports two atomic operations: DOWN and UP. If the semaphore's value is larger than zero, DOWN decrements the semaphore and returns to its caller; otherwise, DOWN releases its processor until another thread increases the semaphore using UP. When control returns to the thread that originally issued the DOWN operation, that thread retries the DOWN operation. [Ch. 5]

sequence coordination—A coordination constraint among threads: for correctness, a certain event in one thread must precede some other certain event in another thread. [Ch. 5]

sequencer—A special type of shared variable used for sequence coordination. The primary operation on a sequencer is TICKET, which operates likes the "take a number" machine in a bakery or post office: two threads concurrently calling TICKET on the same sequencer receive different values, and the ordering of the values returned corresponds to the time ordering of the execution of TICKET. [Ch. 5]

serial transmission—A scheme for increasing the data rate between two modules by sending a series of self-clocking bits over a single transmission line with infrequent or no acknowledgments. [Ch. 7]

serializable—A property of before-or-after actions, that even if several operate concurrently, the result is the same as if they had acted one at a time, in some sequential (in other words, serial) order. [Ch. 9]

server—A module that implements a service. More than one server might implement the same service, or collaborate to implement a fault tolerant version of the service such that even if a server fails, the service is still available. [Ch. 4]

service—A module that responds to actions initiated by clients. [Ch. 4] At the end-to-end layer of a network, the end that responds to actions initiated by the other end. Compare with *client*. [Ch. 7]

set up—The steps required to allocate storage space for and initialize the state of a connection. [Ch. 7]

shadow copy—A working copy of an object that an all-or-nothing action creates so that it can make several changes to the object while the original remains unmodified. When the all-or-nothing action has made all of the changes, it then carefully exchanges the working copy with the original, thus preserving the appearance that all of the changes occurred atomically. Depending on the implementation, either the original or the working copy may be identified as the "shadow" copy, but the technique is the same in either case. [Ch. 9]

shared-secret cryptography—A key-based cryptographic transformation in which the cryptographic key for transforming can be easily determined from the key for the reverse transformation, and vice versa. In most shared-secret systems, the keys for a transformation and its reverse transformation are identical. [Ch. 11]

shared-secret key—The key used by a shared-secret cryptography system. [Ch. 11]

sharing—Allowing an object to be used by more than one other object without requiring multiple copies of the first object. [Ch. 2]

sign—To generate an authentication tag by transforming a message so that a receiver can use the tag to verify that the message is authentic. The word "sign" is usually restricted to public-key authentication systems. The corresponding description for shared-secret authentication systems is "generate a MAC". [Ch. 11]

simple locking—A locking protocol for creating before-or-after actions requiring that no data be read or written before reaching the lock point. For the atomic action to also be all-or-nothing, a further requirement is that no locks be released before commit (or abort). Compare with *two-phase locking*. [Ch. 9]

simple serialization—An atomicity protocol requiring that each newly created atomic action must wait to begin execution until all previously started atomic actions are no longer pending. [Ch. 9]

simplex—Describes a link between two stations that can be used in only one direction. Compare with *duplex*, *half-duplex*, and *full-duplex*. [Ch. 7]

single-acquire protocol—A simple protocol for locking: a thread can acquire a lock only if some other thread has not already acquired it. [Ch. 5]

single-event upset—A synonym for *transient fault*. [Ch. 8]

slave—In a multiple-site replication scheme, a site that takes update requests from only the master site. Compare with *master*. [Ch. 10]

sliding window—In flow control, a technique in which the receiver sends an additional window allocation before it has fully consumed the data from the previous allocation, intending that the new allocation arrive at the sender in time to keep data flowing smoothly, taking into account the transit time of the network. [Ch. 7]

snoopy cache—In a multiprocessor system with a bus and a cache in each processor, a

cache design in which the cache actively monitors traffic on the bus to watch for events that invalidate cache entries. [Ch. 10]

soft modularity—Modularity defined by convention but not enforced by physical constraints. Compare with **enforced modularity**. [Ch. 4]

soft real-time scheduler—A real-time scheduler in which missing a deadline occasionally is acceptable. [Ch. 6]

soft state—State of a running program that the program can easily reconstruct if it becomes necessary to abruptly terminate and restart the program. [Ch. 8]

source—The network attachment point that originated the payload of a packet. Sometimes used as shorthand for *source address*. [Ch. 7]

source address—An identifier of the source of a packet, usually carried as a field in the header of the packet. [Ch. 7]

spatial locality—A kind of locality of reference in which the reference string contains clusters of references to adjacent or nearby addresses. [Ch. 6]

speaks for—A phrase used to express delegation relationships between principals. "A speaks for B" means that B has delegated some authority to A. [Ch. 11]

speculation—A technique to improve performance by performing an operation in advance of receiving a request on the chance that it will be requested. The hope is that the result can be delivered with less latency and with less setup overhead. Examples include demand paging with larger pages than strictly necessary, prepaging, prefetching, and writing dirty pages before the primary device space is needed. [Ch. 6]

spin loop—A situation in which a thread waits for an event to happen without releasing the processor. [Ch. 5]

stability—A property of an object that, once it has a value, it maintains that value indefinitely. Compare with *durability* and *persistence,* terms that have different technical definitions, as explained in Sidebar 2.7. [Ch. 2]

stable binding—A binding that is guaranteed to map a name to the same value for the lifetime of the name space. One of the features of a unique identifier name space. [Ch. 2]

stack algorithm—A class of page-removal algorithms in which the set of pages in a primary device of size m is always a subset of the set of pages in a primary device of size n, if m is smaller than n. Stack algorithms have the property that increasing the size of the memory is guaranteed not to result in increased numbers of missing-page exceptions. [Ch. 6]

starvation—An undesirable situation in which several threads are competing for a shared resource and because of adverse scheduling one or more of the threads never receives a share of the resource. [Ch. 6]

static routing—A method for setting up forwarding tables in which, once calculated, they do not automatically change in response to changes in network topology and load. Compare with *adaptive routing*. [Ch. 7]

static scope—An example of an explicit context, used to resolve names of program variables

in some programming languages. The name resolver searches for a binding starting with the procedure that used the name, then in the procedure in which the first procedure was defined, and so on. Sometimes called *lexical scope*. Compare with *dynamic scope*. [Ch. 2]

station—A device that can send or receive data over a communication link. [Ch. 7]

stop and wait—A synonym for **lock step.** [Ch. 7]

storage—Another term for memory. Memory devices that are non-volatile and are read and written in large blocks are traditionally called storage devices, but there are enough exceptions that in practice the words "memory" and "storage" should be treated as synonyms. [Ch. 2]

store and forward—A forwarding network organization in which transport-layer messages are buffered in a non-volatile memory such as magnetic disk, with the goal that they never be lost. Many authors use this term for any forwarding network. [Ch. 7]

stream—A sequence of data bits or messages that an application intends to flow between two attachment points of a network. It also usually intends that the data of a stream be delivered in the order in which it was sent, and that there be no duplication or omission of data. [Ch. 7]

strict consistency—An interface requirement that temporary violation of a data invariant during an update never be visible outside of the action doing the update. One feature of the read/write coherence memory model is strict consistency. Sometimes called *strong consistency.* [Ch. 10]

stub— A procedure that hides from the caller that the callee is not invoked with the ordinary procedure call conventions. The stub may marshal the arguments into a message and send the message to a service, where another stub unmarshals the message and invokes the callee. [Ch. 4]

supermodule—A set of replicated modules interconnected in such a way that it acts like a single module. [Ch. 8]

supervisor call instruction (SVC)—A processor instruction issued by user modules to pass control of the processor to the kernel. [Ch. 5]

swapping—A feature of some virtual memory systems in which a multilevel memory manager removes a complete address space from a primary device and moves in a complete new one. [Ch. 6]

synonym—One of multiple names that map to the same value. Compare with *alias,* a term that usually, but not always, has the same meaning. [Ch. 2]

system—A set of interconnected components that has an expected behavior observed at the interface with its environment. Contrast with *environment.* [Ch. 1]

tail drop—A strategy for managing an overloaded resource: the system refuses service to the queue entry that arrived most recently. [Ch. 7]

tear down—The steps required to reset the state of a connection and deallocate the space that was used for storage of that state. [Ch. 7]

temporal locality—A kind of locality of reference in which the reference string contains closely-spaced references to the same address. [Ch. 6]

thrashing—An undesirable situation in which the primary device is too small to run a thread or a group of threads, leading to frequent missing-page exceptions. [Ch. 6]

thread—An abstraction that encapsulates the state of a running module. This abstraction encapsulates enough of the state of the interpreter that executes the module so that one can stop a thread at any point in time and later resume it. The ability to stop a thread and resume it later allows virtualization of the interpreter. [Ch. 5]

thread manager—A module that implements the thread abstraction. It typically provides calls for creating a thread, destroying it, allowing the thread to yield, and coordinating with other threads. [Ch. 5]

threat—A potential security violation from either a planned attack by an adversary or an unintended mistake by a legitimate user. [Ch. 11]

throughput—a measure of the rate of useful work done by a service for a given workload. [Ch. 6]

ticket system—A security system in which each principal maintains a list of capabilities, one for each object to which the principal is authorized to have access. [Ch. 11]

tolerated error—An error or class of errors that is both detectable and maskable, and for which a systematic recovery procedure has been implemented. Compare with *detectable error*, *maskable error*, and *untolerated error*. [Ch. 8]

tombstone—A piece of data that will probably never be used again but cannot be discarded because there is still a small chance that it will be needed. [Ch. 7]

trailer—Information that a protocol layer adds to the end of a packet. [Ch. 7]

transaction—A multistep action that is both atomic in the face of failure and atomic in the face of concurrency. That is, it is both all-or-nothing and before-or-after. [Ch. 9]

transactional memory—A memory model in which multiple references to primary memory are both all-or-nothing and before-or-after. [Ch. 9]

transient fault—A fault that is temporary and for which retry of the putatively failed component has a high probability of finding that it is okay. Sometimes called a *single-event upset*. Compare with *persistent fault* and **intermittent fault**. [Ch. 8]

transit time—In a forwarding network, the total delay time required for a packet to go from its source to its destination. In other contexts, this kind of delay is sometimes called *latency*. [Ch. 7]

transmission delay—In a communication network, the component of overall delay contributed by the time spent sending a frame at the available data rate. [Ch. 7]

transport protocol—An end-to-end protocol that moves data between two attachment points of a network while providing a particular set of specified assurances. It can be thought of as a prepackaged set of improvements on the best-effort specification of the network layer. [Ch. 7]

triple-modular redundancy (TMR)—*N*-modular redundancy with $N = 3$. [Ch. 8]

trusted computing base (TCB)—That part of a system that must work properly to make the overall system secure. [Ch. 11]

trusted intermediary—A service that acts as the trusted third party on behalf of multiple, perhaps distrustful, clients. It enforces modularity, thereby allowing multiple distrustful clients to share resources in a controlled manner. [Ch. 4]

two generals dilemma—An intrinsic problem that no finite protocol can guarantee to simultaneously coordinate state values at two places that are linked by an unreliable communication network. [Ch. 9]

two-phase commit—A protocol that creates a higher-layer transaction out of separate, lower-layer transactions. The protocol first goes through a preparation (sometimes called voting) phase, at the end of which each lower-layer transaction reports either that it cannot perform its part or that it is prepared to either commit or abort. It then enters a commitment phase in which the higher-layer transaction, acting as a coordinator, makes a final decision—thus the name two-phase. Two-phase commit has no connection with the similar-sounding term *two-phase locking*. [Ch. 9]

two-phase locking—A locking protocol for before-or-after atomicity that requires that no locks be released until all locks have been acquired (that is, there must be a lock point). For the atomic action to also be all-or-nothing, a further requirement is that no locks for objects to be written be released until the action commits. Compare with *simple locking*. Two-phase locking has no connection with the similar-sounding term *two-phase commit*. [Ch. 9]

undo action—An application-specified action that, when executed during failure recovery or an abort procedure, reverses the effect of some previously performed, but not yet committed, component action. The goal is that neither the original action nor its reversal be visible above the layer that implements the action. Compare with *redo* and *compensate*. [Ch. 9]

unique identifier name space—A name space in which each name, once it is bound to a value, can never be reused for a different value. A unique identifier name space thus provides a stable binding. In a billing system, customer account numbers usually constitute a unique identifier name space. [Ch. 2]

universal name space—A name space of a naming scheme that has only one context. A universal name space has the property that no matter who uses a name it has the same binding. Computer file systems typically provide a universal name space for absolute path names. [Ch. 2]

universe of values—The set of all possible values that can be named by a particular naming scheme. [Ch. 2]

unlimited name space—A name space in which names never have to be reused. [Ch. 3]

untolerated error—An error or class of errors that is undetectable, unmaskable, or unmasked and therefore can be expected to lead to a failure. Compare with *detectable*

error, *maskable error*, and *tolerated error*. [Ch. 8]

user-dependent binding—A binding for which a name used by a shared object resolves to different values, depending on the identity of the user of the shared object. [Ch. 2]

user mode—A feature of a processor that, when set, disallows the use of certain processor features (e.g., changing the page-map address register). Compare with *kernel mode*. [Ch. 5]

utilization—The percentage of capacity used for a given workload. [Ch. 6]

value—The thing to which a name is bound. A value may be a real, physical object, or it may be another name either from the original name space or from a different name space. [Ch. 2]

valid construction—The term used by software designers for *fault avoidance*. [Ch. 8]

version history—The set of all values for an object or variable that have ever existed, stored in journal storage. [Ch. 9]

virtual address—An address that must be translated to a physical address before using it to refer to memory. Compare with *physical address*. [Ch. 5]

virtual circuit—A connection intended to carry a stream through a forwarding network, in some ways simulating an electrical circuit. [Ch. 7]

virtual machine—A method of emulation in which, to maximize performance, a physical processor is used as much as possible to implement virtual instances of itself. [Ch. 5]

virtual machine monitor—The software that implements virtual machines. [Ch. 5]

virtualization—A technique that simulates the interface of a physical object, in some cases creating several virtual objects using one physical instance, in others creating one large virtual object by aggregating several smaller physical instances, and in yet other cases creating a virtual object from a different kind of physical object. [Ch. 5]

virtual memory manager—A memory manager that implements virtual addresses, resolving them to physical addresses by using, for example, a page map. [Ch. 5]

volatile memory—A kind of memory in which the mechanism of retaining information actively consumes energy. When one disconnects the power source it forgets its information content. Compare with *non-volatile memory*. [Ch. 2]

voter—A device used in some NMR designs to compare the output of several nominally identical replicas that all have the same input. [Ch. 8]

well-known name (or address)—A name or address that has been advertised so widely that one can depend on it not changing for the lifetime of the value to which it is bound. In the United States, the emergency telephone number "911" is a well-known name. In some file system designs, sector or block number 1 of every storage device is reserved as a place to store device data, making "1" a well-known address in that context. [Ch. 2]

window—In flow control, the quantity of data that the receiving side of a transport protocol is prepared to accept from the sending side. [Ch. 7]

witness—A (usually cryptographically strong) hash value that attests to the content of a file. Another widely used term for this concept is **fingerprint**. [Ch. 10]

working directory—In a file system, a directory used as a default context, for resolution of relative path names. [Ch. 2]

working set—The set of all addresses to which a thread refers in the interval Δt. If the application exhibits locality of reference, this set of addresses will be small compared to the maximum number of possible addresses during Δt. [Ch. 6]

write-ahead-log (WAL) protocol—A recovery protocol that requires appending a log record in journal storage before installing the corresponding data in cell storage. [Ch. 9]

write tearing—See *atomic storage*.

write-through—A property of a cache: a write operation updates the value in both the primary device and the secondary device before acknowledging completion of the write. (A cache without the write-through property is sometimes called a *write-behind cache*.) [Ch. 6]