- The `InformRequest` PDU is used by a managing entity to notify another managing entity of MIB information that is remote to the receiving entity. The receiving entity replies with a `Response` PDU with the "noError" error status to acknowledge receipt of the `InformRequest` PDU.
- The final type of SNMPv2 PDU is the trap message. Trap messages are generated asynchronously; that is, they are *not* generated in response to a received request but rather in response to an event for which the managing entity requires notification. RFC 3418 defines well-known trap types that include a cold or warm start by a device, a link going up or down, the loss of a neighbor, or an authentication failure event. A received trap request has no required response from a managing entity.

Given the request-response nature of SNMPv2, it is worth noting here that although SNMP PDUs can be carried via many different transport protocols, the SNMP PDU is typically carried in the payload of a UDP datagram. Indeed, RFC 3417 states that UDP is "the preferred transport mapping." Since UDP is an unreliable transport protocol, there is no guarantee that a request, or its response, will be received at the intended destination. The request ID field of the PDU is used by the managing entity to number its requests to an agent; an agent's response takes its request ID from that of the received request. Thus, the request ID field can be used by the managing entity to detect lost requests or replies. It is up to the managing entity to decide whether to retransmit a request if no corresponding response is received after a given amount of time. In particular, the SNMP standard does not mandate any particular procedure for retransmission, or even if retransmission is to be done in the first place. It only requires that the managing entity "needs to act responsibly in respect to the frequency and duration of retransmissions." This, of course, leads one to wonder how a "responsible" protocol should act!

9.3.4 Security and Administration

The designers of SNMPv3 have said that "SNMPv3 can be thought of as SNMPv2 with additional security and administration capabilities" [RFC 3410]. Certainly, there are changes in SNMPv3 over SNMPv2, but nowhere are those changes more evident than in the area of administration and security. The central role of security in SNMPv3 was particularly important, since the lack of adequate security resulted in SNMP being used primarily for monitoring rather than control (for example, `SetRequest` is rarely used in SNMPv1).

As SNMP has matured through three versions, its functionality has grown but so too, alas, has the number of SNMP-related standards documents. This is evidenced by the fact that there is even now an RFC [RFC 3411] that "describes an architecture for describing SNMP Management Frameworks"! While the notion of an "architecture" for "describing a framework" might be a bit much to wrap one's mind around, the goal of RFC 3411 is an admirable one—to introduce a common language for describing the functionality and actions taken by an SNMPv3 agent or

managing entity. The architecture of an SNMPv3 entity is straightforward, and a tour through the architecture will serve to solidify our understanding of SNMP.

So-called **SNMP applications** consist of a command generator, notification receiver, and proxy forwarder (all of which are typically found in a managing entity); a command responder and notification originator (both of which are typically found in an agent); and the possibility of other applications. The command generator generates the `GetRequest`, `GetNextRequest`, `GetBulkRequest`, and `SetRequest` PDUs that we examined in Section 9.3.3 and handles the received responses to these PDUs. The command responder executes in an agent and receives, processes, and replies (using the `Response` message) to received `GetRequest`, `GetNextRequest`, `GetBulkRequest`, and `SetRequest` PDUs. The notification originator application in an agent generates `Trap` PDUs; these PDUs are eventually received and processed in a notification receiver application at a managing entity. The proxy forwarder application forwards request, notification, and response PDUs.

A PDU sent by an SNMP application next passes through the SNMP "engine" before it is sent via the appropriate transport protocol. Figure 9.5 shows how a PDU generated by the command generator application first enters the dispatch module,

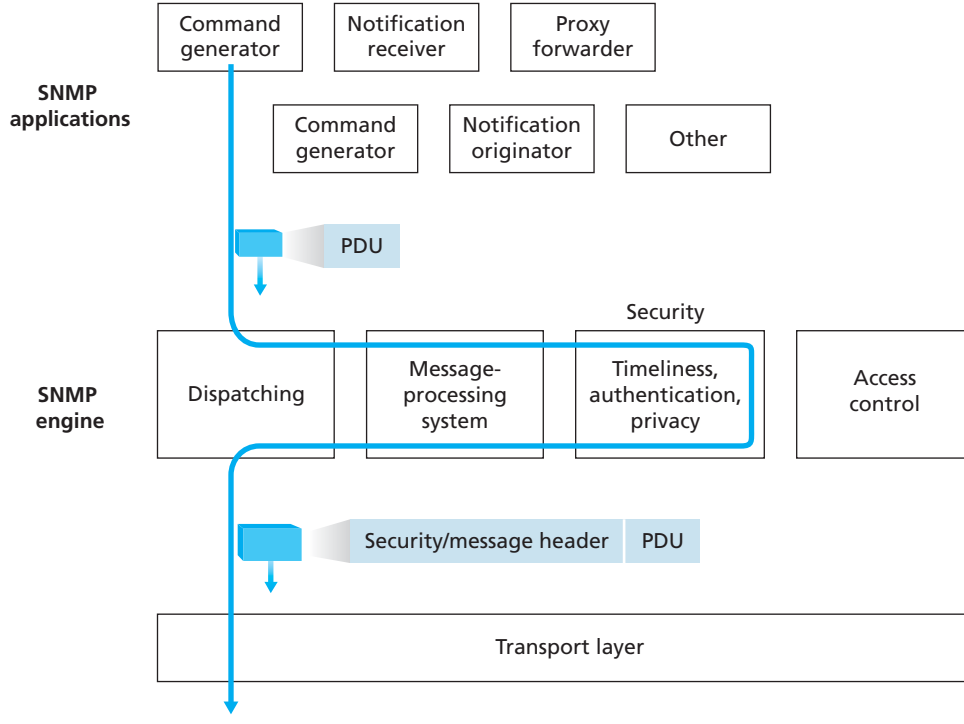

Figure 9.5 ♦ SNMPv3 engine and applications

where the SNMP version is determined. The PDU is then processed in the message-processing system, where the PDU is wrapped in a message header containing the SNMP version number, a message ID, and message size information. If encryption or authentication is needed, the appropriate header fields for this information are included as well; see [RFC 3411] for details. Finally, the SNMP message (the application-generated PDU plus the message header information) is passed to the appropriate transport protocol. The preferred transport protocol for carrying SNMP messages is UDP (that is, SNMP messages are carried as the payload in a UDP datagram), and the preferred port number for the SNMP is port 161. Port 162 is used for trap messages.

We have seen above that SNMP messages are used not just to monitor, but also to control (for example, through the `SetRequest` command) network elements. Clearly, an intruder that could intercept SNMP messages and/or generate its own SNMP packets into the management infrastructure could wreak havoc in the network. Thus, it is crucial that SNMP messages be transmitted securely. Surprisingly, it is only in the most recent version of SNMP that security has received the attention that it deserves. SNMPv3 security is known as **user-based security** [RFC 3414] in that there is the traditional concept of a user, identified by a username, with which security information such as a password, key value, or access privileges are associated. SNMPv3 provides for encryption, authentication, protection against playback attacks (see Section 8.3), and access control.

- *Encryption.* SNMP PDUs can be encrypted using the Data Encryption Standard (DES) in Cipher Block Chaining (CBC) mode. Note that since DES is a shared-key system, the secret key of the user encrypting data must be known by the receiving entity that must decrypt the data.
- *Authentication.* SNMP uses the Message Authentication Code (MAC) technique that we studied in Section 8.3.1 to provide both authentication and protection against tampering [RFC 4301]. Recall that a MAC requires the sender and receiver both to know a common secret key.
- *Protection against playback.* Recall from our discussion in Chapter 8 that nonces can be used to guard against playback attacks. SNMPv3 adopts a related approach. In order to ensure that a received message is not a replay of some earlier message, the receiver requires that the sender include a value in each message that is based on a counter in the *receiver.* This counter, which functions as a nonce, reflects the amount of time since the last reboot of the receiver's network management software and the total number of reboots since the receiver's network management software was last configured. As long as the counter in a received message is within some margin of error of the receiver's actual value, the message is accepted as a nonreplay message, at which point it may be authenticated and/or decrypted. See [RFC 3414] for details.
- *Access control.* SNMPv3 provides a view-based access control [RFC 3415] that controls which network management information can be queried and/or set by which users. An SNMP entity retains information about access rights and