UNIT V

APPLICATION LAYER

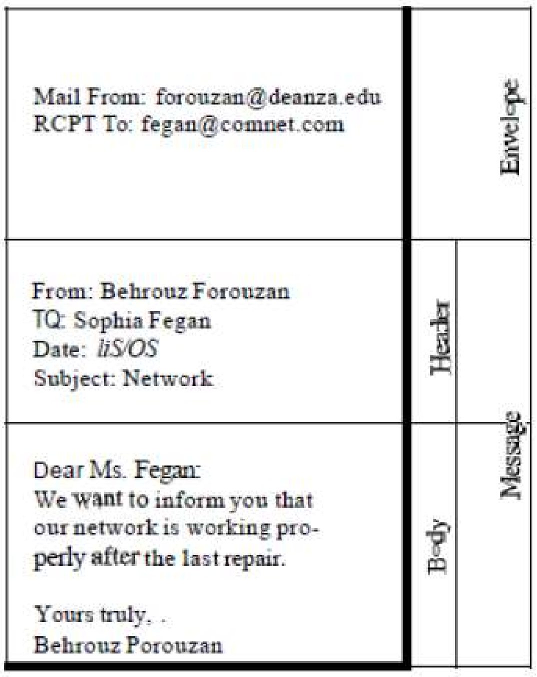
Traditional applications -Electronic Mail (SMTP, POP3, IMAP, MIME) – HTTP – Web Services – DNS - SNMP

5.1. TRADITIONAL APPLICATIONS

- Applications have become feasible recently: streaming applications (e.g., multimedia applications like video and audio) and various overlay based applications.
- It is important to distinguish between application *programs* and application *protocols*.
- For example, the Hyper Text Transport Protocol (HTTP) is an application protocol that is used to retrieve Web pages from remote servers. Many different application programs are web clients like Internet Explorer, Chrome, Firefox, and Safari.
- There are two very widely used, standardized application protocols:
  - ✓ Simple Mail Transfer Protocol (SMTP) is used to exchange electronic mail.
  - ✓ Hyper Text Transport Protocol (HTTP) is used to communicate between web browsers and web servers.

5.1.1. ELECTRONIC MAIL

- Email is one of the oldest network applications.
- Email was a key application when the network was created—remote access to computing resources was the main design goal—but it turned out to be a useful application that continues to be extremely popular.


Mail From: forouzan@deanza.edu
RCPT To: fegan@comnet.com
Envelope
From: Behrouz Forouzan
TO: Sophia Fegan
Date: liS/OS
Subject: Network
Header
Dear Ms. Fegan:
We want to inform you that our network is working properly after the last repair.
Yours truly, .
Behrouz Porouzan
Body
Message
Figure 5.1 Electronic Mail

A. MIME

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail.

<u>Message Format</u>

- RFC 822 defines messages to have two parts: a *header* and a *body*. Both parts are represented in ASCII text.
- The header is separated from the message body by a blank line.
- Each header line contains a type and value separated by a colon.
- The message header is a series of <CRLF>-terminated lines.
- Many of these header lines are familiar to users, since they are asked to fill them out when they compose an email message; for example, the To: header identifies the message recipient, and the Subject: header says something about the purpose of the message. Other headers are filled in by the underlying mail delivery system. Examples include Date: (when the message was transmitted), From: (what user sent the message), and Received: (each mail server that handled this message.
- RFC 822 allows email messages to carry many different types of data :audio, video, images, PDF documents, and so on.
- MIME consists of three basic pieces.

- The first piece is a collection of header lines that augment the original set defined by RFC 822. They include MIME-Version, Content-Description, Content-Type and Content-Transfer-Encoding.
- The second piece is definitions for a set of content types and subtypes. MIME also defines a multipart type that says how a message carrying more than one data type is structured.
- The third piece is a way to encode the various data types. It is important that email messages contain only ASCII, because they might pass through a number of intermediate systems.

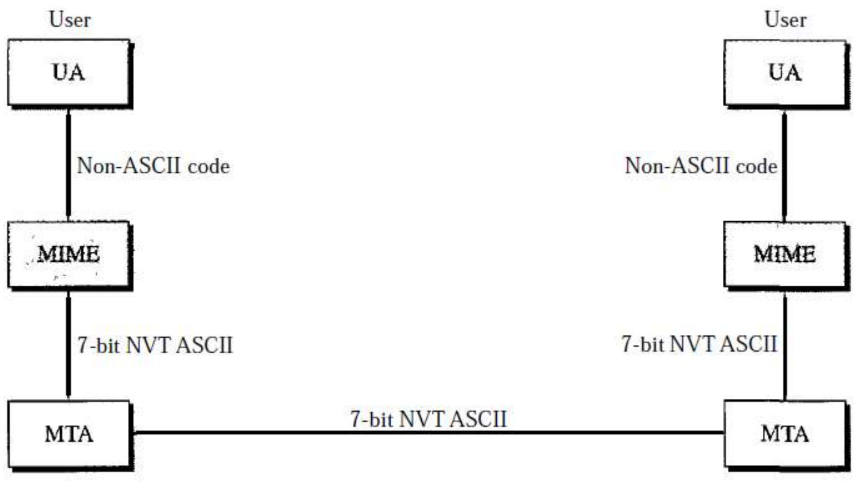


Figure 5.2 MIME

➢ Example

```
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="-------417CA6E2DE4ABCAFBC5"
From: Alice Smith <Alice@cisco.com>
To: Bob@cs.Princeton.edu
Subject: promised material
Date: Mon, 07 Sep 1998 19:45:19 -0400
---------417CA6E2DE4ABCAFBC5
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Bob,
Here's the jpeg image and draft report I promised.
--Alice
---------417CA6E2DE4ABCAFBC5
Content-Type: image/jpeg
Content-Transfer-Encoding: base64
         ... unreadable encoding of a jpeg figure
---------417CA6E2DE4ABCAFBC5
Content-Type: application/postscript; name="draft.ps"
Content-Transfer-Encoding: 7bit
         ... readable encoding of a PostScript document
```

B. SMTP

Message Transfer

- The majority of email was moved from host to host using only SMTP. While SMTP continues to play a central role, it is now just one email protocol of several, Internet Message Access Protocol (IMAP) and Post Office Protocol (POP) being two other important protocols for retrieving mail messages.
- Users interact with a *mail reader* when they compose, file, search, and read their email.
- Users remotely access their mailbox from their laptop or smartphone; they do not first log into the host that stores their mail (a mail server).
- A second mail transfer protocol, such as POP or IMAP, is used to remotely download email from a mail server to the user's device.
- There is a *mail daemon* (or process) running on each host that holds a mailbox.
- A process, called a *message transfer agent* (MTA), as playing the role of a post office: Users give the daemon messages they want to send to other users, the daemon uses SMTP running over TCP to transmit the message to a daemon running on another machine, and the daemon puts incoming messages into the user's mailbox (where that user's mail reader can later find them).
- The MTA on a sender's machine establishes an SMTP/TCP connection to the MTA on the recipient's mail server, in many cases the mail traverses one or more *mail gateways* on its route from the sender's host to the receiver's host.
- Like the end hosts, these gateways also run a message transfer agent process,the intermediate nodes are called *gateways* since their job is tostore and forward email messages, much like an "IP gateway" stores and forwards IP datagrams.
- The only difference is that a mail gateway typically buffers messages on disk and retransmits them to the next machine while an IP router buffers datagrams in memory and retry transmitting them for a fraction of a second.
- Example

```
HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]
MAIL FROM:<Bob@cs.princeton.edu>
250 OK
RCPT TO:<Alice@cisco.com>
250 OK
RCPT TO:<Tom@cisco.com>
550 No such user here
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
      ...etc. etc. etc.
HELO cs.princeton.edu
250 Hello daemon@mail.cs.princeton.edu [128.12.169.24]
```

```
MAIL FROM:<Bob@cs.princeton.edu>
250 OK
RCPT TO:<Alice@cisco.com>
250 OK
RCPT TO:<Tom@cisco.com>
550 No such user here
DATA
354 Start mail input; end with <CRLF>.<CRLF>
Blah blah blah...
...etc. etc. etc.
```

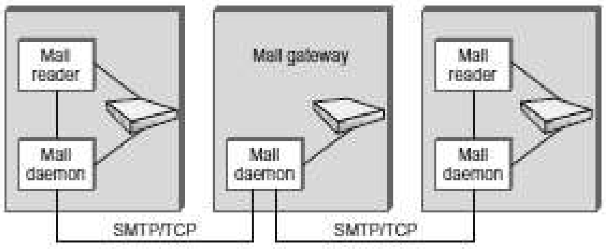


Figure 5.3 Sequence of mail gateways store and forward e-mail messages

C. IMAP

Mail Reader

- The final step is for the user to actually retrieve his or her messages from the mailbox, read them, reply to them, and possibly save a copy for future reference.
- The user performs all these actions by interacting with a mail reader.
- The reader was originally just a program running on the same machine as the user's mailbox, in which case it could simply read and write the file that implements the mailbox.
- Today, the user accesses his or her mailbox from a remote machine using protocol,such as POP or IMAP.
- IMAP is similar to SMTP in many ways. It is a client/server protocol running over TCP, where the client issues commands in the form of <CRLF>-terminated ASCII text lines and the mail server responds in kind.
- The exchange begins with the client authenticating him- or herself and identifying the mailbox he or she wants to access.
- This can be represented by the simple state transition diagram shown in Figure 5.4. In this diagram, LOGIN, AUTHENTICATE, SELECT, EXAMINE, CLOSE, and

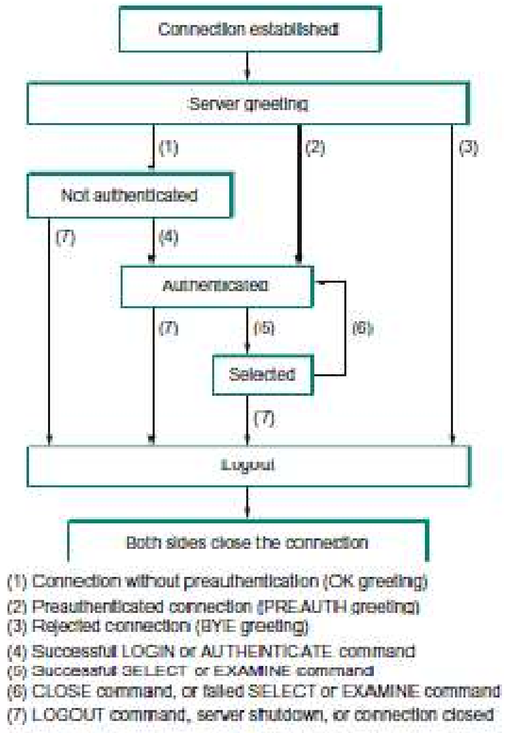

Connection established
Server greeting
(1)
(2)
(3)
Not authenticated
(7)
(4)
Authenticated
(7)
(5)
(6)
Selected
(7)
Logout
Both sides close the connection
(1) Connection without preauthentication (OK greeting)
(2) Preauthenticated connection (PREAUTH greeting)
(3) Rejected connection (BYE greeting)
(4) Successful LOGIN or AUTHENTICATE command
(5) Successful SELECT or EXAMINE command
(6) CLOSE command, or failed SELECT or EXAMINE command
(7) LOGOUT command, server shutdown, or connection closed

Figure 5.4 IMAP State Transition Diagram

LOGOUT are example commands that the client can issue, while OK is one possible server response.

- Other common commands include FETCH, STORE, DELETE, and EXPUNGE, with the obvious meanings.
- Additional server responses include NO (client does not have permission to perform that operation) and BAD (command is ill formed).
- When the user asks to FETCH a message, the server returns it in MIME format and the mail reader decodes it.
- In addition to the message itself, IMAP also defines a set of message *attributes* that are exchanged as part of other commands, independent of transferring the message itself.
- Message attributes include information like the size of the message and, more interestingly, various *flags* associated with the message (e.g., Seen,Answered, Deleted, and Recent). These flags are used to keep the client and server synchronized.
- The client issues an EXPUNGE command to the server, which knows to actually remove all earlier deleted messages from the mailbox.
- Finally, the user replies to a message, or sends a new message.

D. ***POP3***

- Post Office Protocol, version 3 (POP3) is simple and limited in functionality.
- The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.
- Mail access starts with the client when the user needs to download e-mail from the mailbox on the mail server.
- The client opens a connection to the server on TCP port 110.
- It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one. Figure 5.5 shows an example of downloading using POP3.

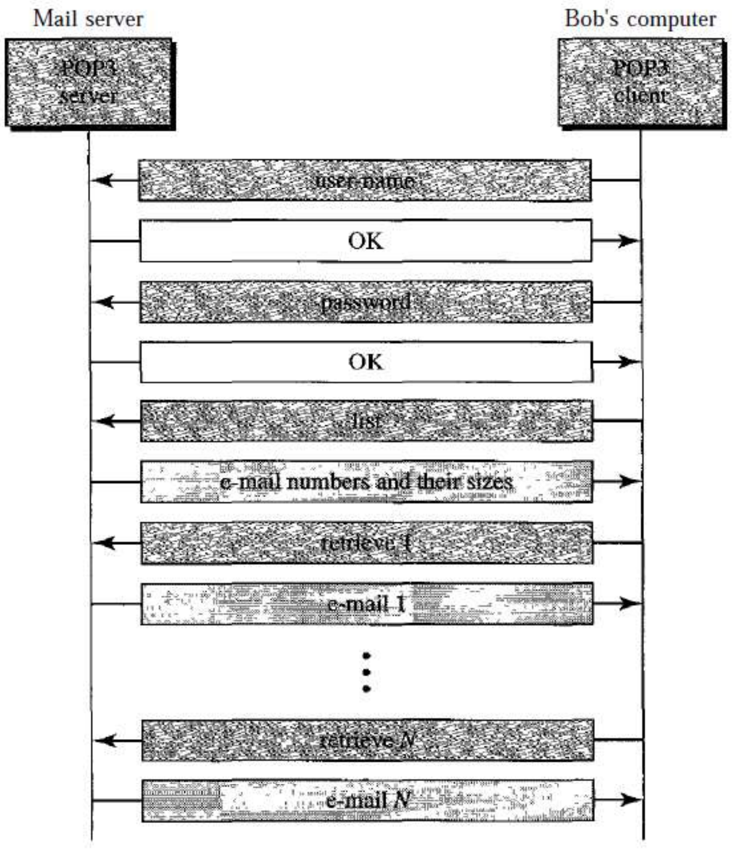


Figure 5.5 The exchange ofcommands and responses in POP3

- POP3 has two modes:
  - ✓ *the delete mode* the mail is deleted from the mailbox after each retrieval. The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying.
  - ✓ *the keep mode* In the keep mode, the mail remains in the mailbox after retrieval. The keep mode is normally used when the user accesses her mail away from her primary computer (e.g., a laptop). The mail is read but kept in the system for later retrieval and organizing.

5.1.2. HTTP:

- The original goal of the Web was to find a way to organize and retrieve information, drawing on ideas about hypertext—interlinked documents—that had been around since at least the 1960s.
- The core idea of hypertext is that one document can link to another document,and the protocol (HTTP) and document language (HTML) were designed to meet that goal.
- Clearly, if we want to organize information into a system of linked documents or objects, we need to be able to retrieve one document to get started. Hence, any web browser has a function that allows the user to obtain an object by opening a URL.
- Uniform Resource Locators (URLs) provide information that allows objects on the Web to be located, and they look like the following: http://www.cs.princeton.edu/index.html
  - If we open a particular URL the web browser would open a TCP connection to the web server at a machine called www.cs.princeton.edu and immediately retrieve and display the file called index.html.
- Most files on the Web contain images and text, and many have other objects such as audio and video clips, pieces of code, etc.
- They also frequently include URLs that point to other files that may be located on other machines, which is the core of the "hypertext" part of HTTP and HTML.
- A web browser has some way in which we can recognize URLs and the browser open them.
- These embedded URLs are called *hypertext links*.
- When we ask the web browser to open one of these embedded URLs (e.g., by pointing and clicking on it with a mouse), it will open a new connection and retrieve and display a new file. This is called *following a link*.
- It thus becomes very easy to hop from one machine to another around the network.
- When we ask your browser to view a page, the browser (the client) fetches the page from the server using HTTP running over TCP.
- Like SMTP,HTTP is a text-oriented protocol.
- At its core, HTTP is a request/response protocol, where every message has the general form

```
START_LINE <CRLF>
MESSAGE_HEADER <CRLF>
<CRLF>
```

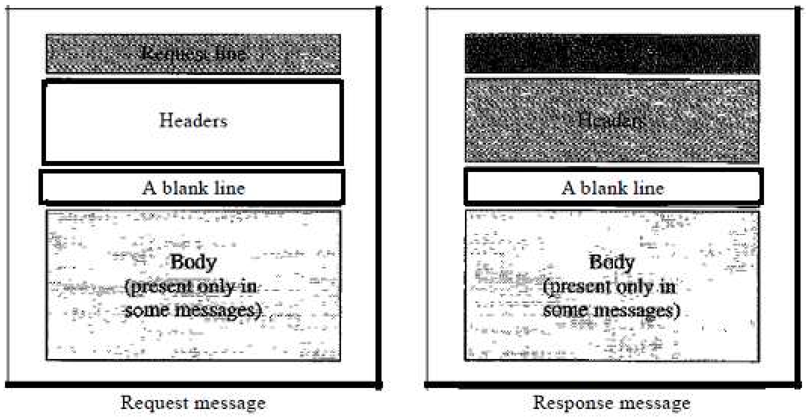


Figure 5.8 HTTP Message Format

```
MESSAGE_BODY <CRLF>
```

- where, as before, <CRLF> stands for carriage-return+line-feed.

- The first line (START LINE) indicates whether this is a request message or a response message. It identifies the "remote procedure" to be executed or the *status* of the request.
- The next set of lines specifies a collection of options and parameters that qualify the request or response.
- There are zero or more of these MESSAGE HEADER lines—the set is terminated by a blank line—each of which looks like a header line in an email message.

➢ HTTP defines many possible header types, some of which pertain to request messages, some to response messages, and some to the data carried in the message body.

Request Messages

➢ The first line of an HTTP request message specifies three things: the operation to be performed, the Web page the operation should be performed on, and the version of HTTP being used.

➢ HTTP defines a wide assortment of possible request operations—including *write* operations that allow a Web page to be posted on a server—the two most common operations are GET (fetch the specified Web page) and HEAD (fetch status information about the specified Web page).

➢ The former is obviously used when the browser wants to retrieve and display a Web page.

➢ The latter is used to test the validity of a hypertext link or to see if a particular page has been modified since the browser last fetched it.

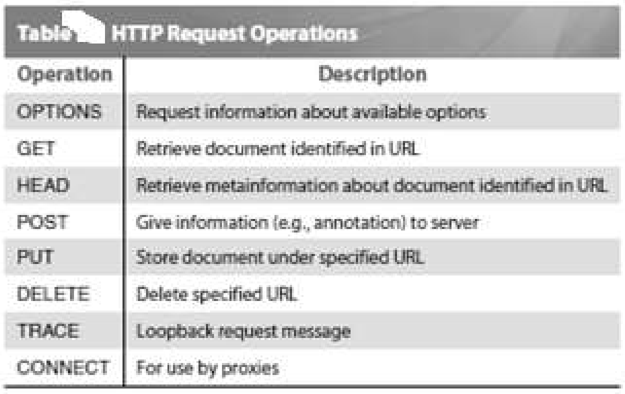


Table HTTP Request Operations

Operation	Description
OPTIONS	Request information about available options
GET	Retrieve document identified in URL
HEAD	Retrieve metainformation about document identified in URL
POST	Give information (e.g., annotation) to server
PUT	Store document under specified URL
DELETE	Delete specified URL
TRACE	Loopback request message
CONNECT	For use by proxies

Response Message

➢ Like request messages, response messages begin with a single START LINE.

➢ In this case, the line specifies the version of HTTP being used, a three-digit code indicating whether or not the request was successful, and a text string giving the reason for the response.

➢ There are five general types of response codes, with the first digit of the code indicating its type. Table summarizes the five types of codes.

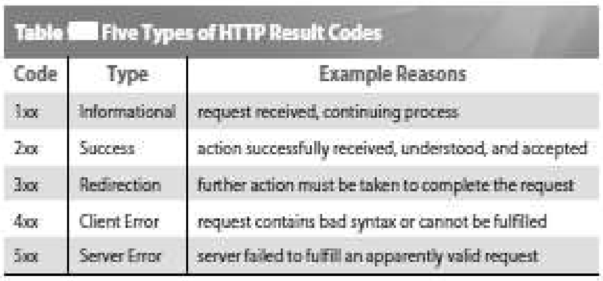


Table Five Types of HTTP Result Codes

Code	Type	Example Reasons
1xx	Informational	request received, continuing process
2xx	Success	action successfully received, understood, and accepted
3xx	Redirection	further action must be taken to complete the request
4xx	Client Error	request contains bad syntax or cannot be fulfilled
5xx	Server Error	server failed to fulfill an apparently valid request

Uniform Resource Identifier

➢ The URLs that HTTP uses as addresses are one type of *Uniform Resource Identifier* (URI).

➢ A URI is a character string that identifies a resource, where a resource can be anything that has identity, such as a document, an image, or a service.

➢ The format of URIs allows various more specialized kinds of resource identifiers to be incorporated into the URI space of identifiers.

➢ The first part of a URI is a *scheme* that names a particular way of identifying a certain kind of resource, such as mail to for email addresses or file for file names.

- The second part of a URI, separated from the first part by a colon, is the *scheme-specific part*. It is a resource identifier consistent with the scheme in the first part.
- A resource doesn't have to be retrievable or accessible.

TCP CONNECTIONS

- The original version of HTTP (1.0) established a separate TCP connection for each data item retrieved from the server.
- Figure 5.6 shows the sequence of events for fetching a page that has just a single embedded object.
- Colored lines indicate TCP messages, while black lines indicate the HTTP requests and responses.
- There are two round trip times are spent setting up TCP connections while another two (at least) are spent getting the page and image
- There is also processing cost on the server to handle the extra TCP connection establishment and termination.
- To overcome this situation, HTTP version 1.1 introduced *persistent connections*—the client and server can exchange multiple request/ response messages over the same TCP connection.
- Persistent connections have many advantages.
  - ❖ They eliminate the connection setup overhead, thereby reducing the load on the server.
  - ❖ A client can send multiple request messages down a single TCP connection, TCP's congestion window mechanism is able to operate more efficiently.

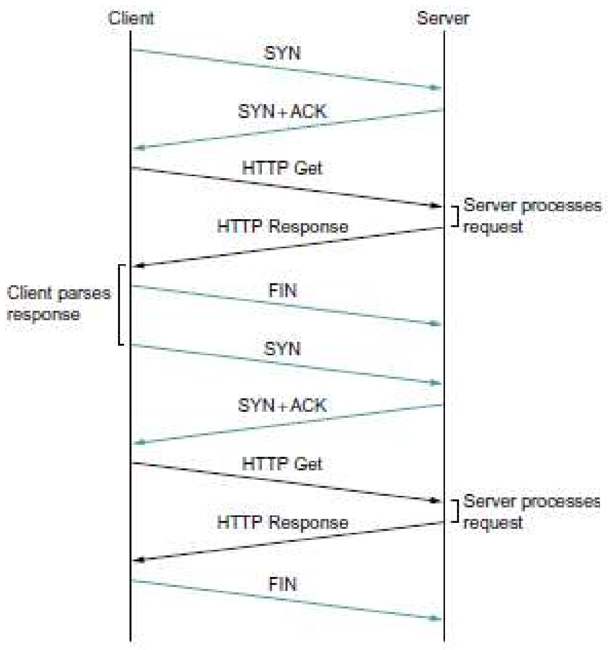


Figure 5.6 HTTP 1.0 behavier

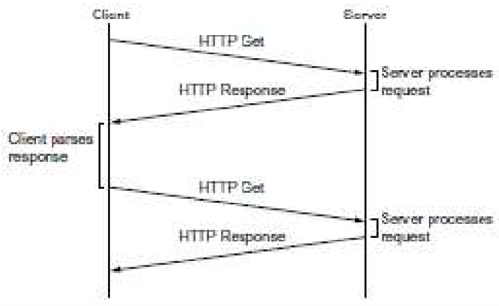


Figure 5.7 HTTP 1.1. behavier with persistent connections

Caching

- Caching has many benefits, from the client's perspective, a page that can be retrieved from a nearby cache can be displayed much more quickly than if it has to be fetched from across the world.
- From the server's perspective, having a cache intercept and satisfy a request reduces the load on the server.
- Caching can be implemented in many different places. For example, a user's browser can cache recently accessed pages and simply display the cached copy if the user visits the same page again.

- As another example, a site can support a single site-wide cache. This allows users to take advantage of pages previously downloaded by other users.
- Internet Service Providers (ISPs) can cache pages.
- The users within the site most likely know what machine is caching pages on behalf of the site, and they configure their browsers to connect directly to the caching host. This node is sometimes called a *proxy.*
- There are a set of *cache directives* that must be obeyed by all caching mechanisms along the request/response chain. These directives specify whether or not a document can be cached, how long it can be cached, how fresh a document must be, and so on.

5.1.3. WEB SERVICES

- There is increasing demand for direct computer to computer interaction.
- Much of the motivation for enabling direct application-to-application communication comes from the business world.
- Historically, interactions between enterprises—businesses or other organizations—have involved some manual steps such as filling out an order form or making a phone call to determine whether some product is in stock.
- Even within a single enterprise it is common to have manual steps between software systems that cannot interact directly because they were developed independently.
- Increasingly, such manual interactions are being replaced with direct application-to-application interaction. Here is a simple example of what we are talking about.
- The protocol specifications and the implementations of those protocols for traditional applications like electronic mail and file transfer have typically been developed by a small group of networking experts.
- To enable the vast number of potential network applications to be developed quickly, it was necessary to come up with some technologies that simplify and automate the task of application protocol design and implementation.
- Two architectures have been advocated as solutions to this problem. Both architectures are called *Web Services*, taking their name from the term for the individual applications that offer a remotely accessible service to client applications to form network applications.
- The terms used as informal shorthand to distinguish the two Web Services architectures are *SOAP* and *REST.*
- The SOAP architecture's approach to the problem is to make it feasible, to generate protocols that are customized to each network application
- The key elements of the approach are a framework for protocol specification, software tool kits for automatically generating protocol implementations from the specifications, and modular partial specifications that can be reused across protocols.
- The REST architecture's approach to the problem is to regard individual Web Services as World Wide Web resources identified by URIs and accessed via HTTP.

Custom Application (WSDL, SOAP)

- The architecture informally referred to as SOAP is based on *Web Services Description Language* (WSDL) and *SOAP*. Both of these standards are issued by the World Wide Web Consortium (W3C).
- This is the architecture that people usually mean when they use the term Web Services without any preceding qualifier.

- WSDL and SOAP are frameworks for specifying and implementing application protocols and transport protocols, respectively.
- They are generally used together, although that is not strictly required.
- WSDL is used to specify application-specific details such as what operations are supported, the formats of the application data to invoke or respond to those operations, and whether an operation involves a response.
- SOAP's role is to make it easy to define a transport protocol with exactly the desired semantics regarding protocol features such as reliability and security.
- Both WSDL and SOAP consist primarily of a protocol specification language.

A. WSDL

Defining Application Protocols WSDL

- WSDL has chosen a procedural *operation* model of application protocols.
- An abstract Web Service interface consists of a set of named operations, each representing a simple interaction between a client and the Web Service.
- An operation is analogous to a remotely callable procedure in an RPC system. An example from W3C's WSDL Primer is a hotel reservation Web Service with two operations, Check Availability and Make Reservation.
- Each operation specifies a *Message Exchange Pattern* (MEP) that gives the sequence in which the messages are to be transmitted, including the fault messages to be sent when an error disrupts the message flow.
- Several MEPs are predefined, and new custom MEPs can be defined, but in practice only two MEPs are being used: **In-Only** (a single message from client to service) and **In-Out** (a request from client and a corresponding reply from service).
- MEPs are templates that have placeholders instead of specific message types or formats, so part of the definition of an operation involves specifying which message formats to map into the placeholders in the pattern.
- Message formats are not defined at the bit level Instead they are defined as an abstract data model using XML Schema.
- XML Schema provides a set of primitive data types and ways to define compound data types.
- Data that conforms to an XML Schema-defined format its abstract data model can be concretely represented using XML, or it can use another representation, such as the "binary" representation.
- WSDL's concrete part specifies an underlying protocol, how MEPs are mapped onto it, and what bit-level representation is used for messages on the wire. This part of a specification is known as a *binding*.
- WSDL has predefined bindings for HTTP and SOAP-based protocols.

B. SOAP

Defining Transport Protocol

- "SOAP provides a simple messaging framework whose core functionality is concerned with providing extensibility."
- SOAP uses many of the same strategies as WSDL, including message formats defined using XML Schema, bindings to underlying protocols, Message Exchange Patterns, and reusable specification elements identified using XML namespaces.
- SOAP is used to define transport protocols with exactly the features needed to support a particular application protocol.

- SOAP aims to make it feasible to define many such protocols by using reusable components.
- Each component captures the header information and logic that go into implementing a particular feature.
- *A SOAP* feature is an extension of the SOAP messaging framework.
- Although SOAP poses no constraints on the potential scope of such features, example features may include "reliability," "security," "correlation," "routing," and message exchange patterns (MEPs) such as request/response, one-way, and peer-to-peer conversations.
- A SOAP feature specification must include:
  - A URI that identifies the feature.
  - The state information and processing, abstractly described, that is required at each SOAP node to implement the feature.
  - The information to be relayed to the next node.
  - If the feature is a MEP, the life cycle and temporal/causal relationships of the messages exchanged—for example, responses follow requests and are sent to the originator of the request.
- There are two strategies for defining a SOAP protocol that will implement them.
- One is by layering: binding SOAP to an underlying protocol in such a way as to derive the features.
- The second and more flexible way to implement features involves *header blocks*.
- A SOAP message consists of an Envelope, which contains a Header that contains header blocks, and a Body, which contains the payload destined for the ultimate receiver.

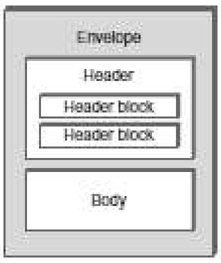


Figure 5.8 SOAP message structure

- A digital signature is used to implement authentication, a sequence number is used for reliability, and a Check sum is used to detect message corruption.
- A SOAP header block is intended to encapsulate the header information that corresponds to a particular feature.

<u>Standardizing Web Services Protocols</u>

- WSDL and SOAP aren't protocols, they are standards for *specifying* protocols. For different enterprises to implement Web Services that interoperate with each other, it is not enough to agree to use WSDL and SOAP to define their protocols.
- The tension between standardization and customization is tackled by establishing partial standards called *profiles*.
- A profile is a set of guidelines that narrow or constrain choices available in WSDL, SOAP, and other standards that may be referenced in defining a protocol.
- They may at the same time resolve ambiguities or gaps in those standards.
- The broadest and most widely adopted profile is known as the *WS-I Basic Profile*.
- It was proposed by the Web Services Interoperability Organization (WS-I), specified by the World Wide Web Consortium (W3C).
- The Basic Profile resolves some of the most basic choices faced in defining a Web Service
- . It also specifies which versions of WSDL and SOAP must be used.
- The *WS-I Basic Security Profile* adds security constraints to the Basic Profile .

- The standards known collectively as *WS-** include WS-Reliability, WS-Reliable- Messaging, WS-Coordination, and WS-Atomic Transaction.

A Generic Application Protocol (REST)

- The WSDL/SOAP Web Services architecture is based on the assumption that the best way to integrate applications across networks is via protocols that are customized to each application. That architecture is designed to make it practical to specify and implement all those protocols.
- The REST Web Services architecture is based on the assumption that the best way to integrate applications across networks is by re-applying a protocol by which they exchange information, along the lines of the simple package tracking.
- This standardization is crucial for tool support as well as interoperability protocols.
- Existing content distribution networks (CDNs) can be used to support REST applications.
- The Web and REST may also have an advantage in resolvability.
- Although the WSDL and SOAP *frameworks* are highly flexible with regard to what new features and bindings can go into the definition of a protocol, that flexibility is irrelevant once the protocol is defined.
- Standardized protocols such as HTTP are designed with a provision for being extended in a backward-compatible way.
- HTTP's own extensibility takes the form of headers, new methods, and new content types.

5.1.4. DNS

- Name services are sometimes called *middleware* because they fill a gap between applications and the underlying network.
- Host names differ from host addresses in two important ways.
- First, they are usually of variable length and mnemonic, thereby making them easier for humans to remember.
- Second, names typically contain no information that helps the network locate the host.
- Addresses have routing information embedded in them; *flat* addresses (those not divisible into component parts) are the exception.
- A *name space* defines the set of possible names.
- A name space can be either *flat* (names are not divisible into components) or *hierarchical.*
- Second, the naming system maintains a collection of *bindings* of names to values.
- The value can be anything we want the naming system to return when presented with a name; in many cases, it is an address.
- A *resolution mechanism* is a procedure that, when invoked with a name, returns the corresponding value.
- The Internet has a particularly well-developed naming system in place—the Domain Name System (DNS)as a framework of naming hosts.
- DNS employs a hierarchical namespace rather than a flat name space, and the "table" of bindings that implements this name space is partitioned into disjoint pieces and distributed throughout the Internet.
- These subtables are made available in name servers that can be queried over the network.

Domain Hierarchy

- DNS implements a hierarchical name space for Internet objects.

- Unlike Unix file names, which are processed from left to right with the naming components separated with slashes, DNS names are processed from right to left and use periods as the separator.D

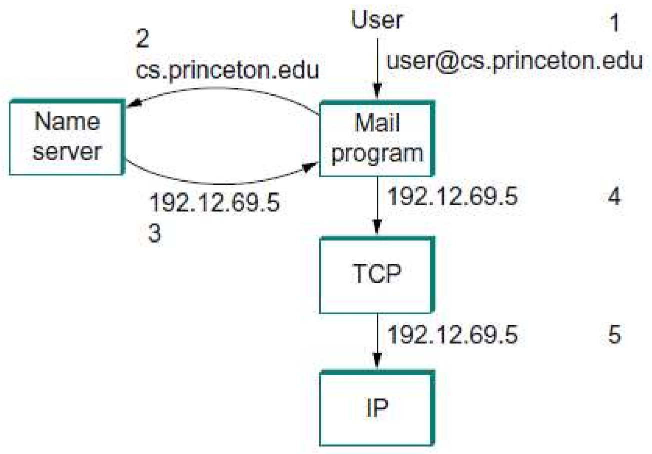


Figure 5.9 Names translated into addresses, where the numbers 1 to 5 show the sequence of steps in the process.

- Domain names are used to name Internet "objects." is that DNS is not strictly used to map host names into host addresses.
- It is more accurate to say that DNS maps domain names into values.
- There are domains for each country, plus the "big six" domains: .edu, .com, .gov, .mil, .org, and .net.
- These six domains were all originally based in the United States (where the Internet and DNS were invented); for example, only U.S.-accredited educational institutions can register an .edu domain name.
- In recent years, the number of top-level domains has been expanded, partly to deal with the high demand for .com domains names.

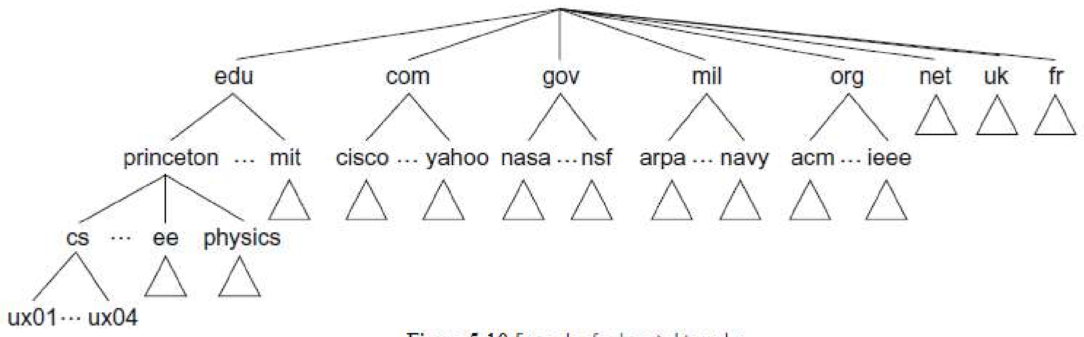


Figure 5.10 Example of a domain hierarchy.

Name Servers

- The complete domain name hierarchy exists only in the abstract.
- The first step is to partition the hierarchy into sub trees called *zones*.
- Figure 5.10 shows how the hierarchy given in Figure 5.11 might be divided into zones.
- Each zone can be thought of as corresponding to some administrative authority that is responsible for that portion of the hierarchy.
- The relevance of a zone is that it corresponds to the fundamental unit of implementation in DNS,the name server.

- The information contained in each zone is implemented in two or more name servers.

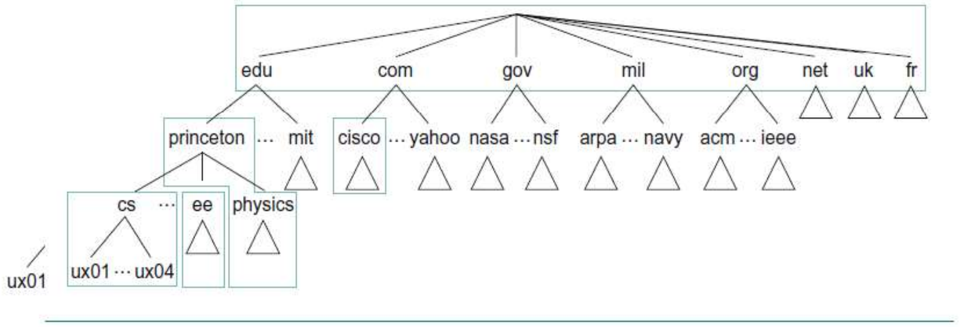


Figure 5.11 Domain hierarchy partitioned into zones.

- Each name server, in turn, is a program that can be accessed over the Internet.
- Clients send queries to name servers, and name servers respond with the requested information.
- Each name server implements the zone information as a collection of *resource records*. A
- A resource record is a name-to-value binding or, more specifically, a 5-tuple that contains the following fields: hName, Value, Type, Class, TTL i.
- Record types include:
  - ❖ NS—The Value field gives the domain name for a host that is running a name server that knows how to resolve names within the specified domain.
  - ❖ CNAME—The Value field gives the canonical name for a particular host; it is used to define aliases.
  - ❖ MX—The Value field gives the domain name for a host that is running a mail server that accepts messages for the specified domain.
- The Class field was included to allow entities other than the NIC to define useful record types.
- The time-to-live (TTL) field shows how long this resource record is valid. It is used by servers that cache resource records from other servers; when the TTL expires, the server must evict the record from its cache.

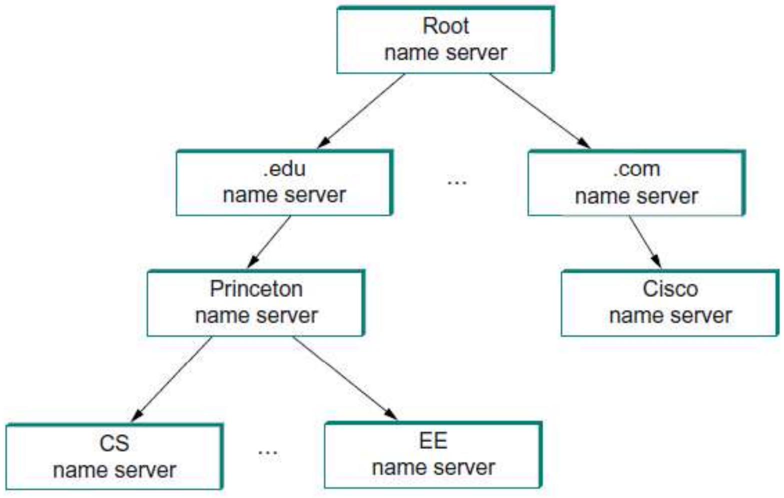


Figure 5.12 Hierarchy of name servers.

Name Resolution

- The client send a query containing this name to one of the root servers.
- The root server, unable to match the entire name, returns the best match it has in the NS record.
- The server also returns all records that are related to this record.
- The client, having not received the answer it was after, next sends the same query to the name server at IP host 192.5.6.32.
- This server also cannot match the whole name and so returns the NS and corresponding records for the princeton.edu domain.
- Once again, the client sends the same query as before to the server at IP host 128.112.129.15, and this time gets back the NS record and corresponding record for the cs.princeton.edu domain.
- This time, the server that can fully resolve the query has been reached.
- A final query to the server at 128.112.136.10 yields the a record for penguins.cs.princeton.edu, and the client learns that the corresponding IP address is 128.112.155.166.

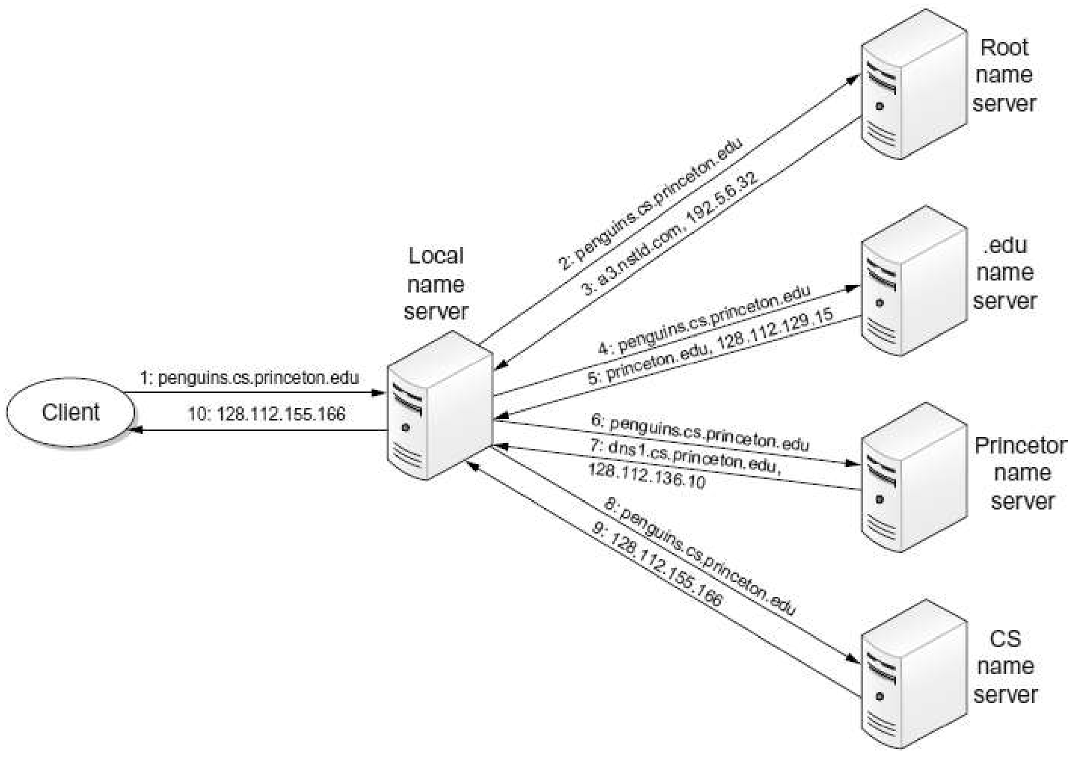

Root name server
.edu name server
Princetor name server
CS name server
Local name server
Client
1: penguins.cs.princeton.edu
10: 128.112.155.166
2: penguins.cs.princeton.edu
3: a3.nstld.com, 192.5.6.32
4: penguins.cs.princeton.edu
5: princeton.edu, 128.112.129.15
6: penguins.cs.princeton.edu
7: dns1.cs.princeton.edu, 128.112.136.10
8: penguins.cs.princeton.edu
9: 128.112.155.166

Figure 5.13 Name resolution in practice, where the numbers 1 to 10 show the sequence of steps in the process.

5.5 SNMP

- A network is a complex system, both in terms of the number of nodes that are involved and in terms of the suite of protocols that can be running on any one node an issue that pervades the entire network architecture.
- Since the nodes we want to keep track of are distributed, our only real option is to use the network to manage the network. This means we need a protocol that allows us to read, and possibly write, various pieces of state information on different network nodes.
- The most widely used protocol for this purpose is the Simple Network Management Protocol (SNMP).

- SNMP is essentially a specialized request/reply protocol that supports two kinds of request messages: GET and SET.
- The former is used to retrieve a piece of state from some node, and the latter is used to store a new piece of state in some node.
- An SNMP server running on that node receives the request, locates the appropriate piece of information, and returns it to the client program, which then displays it to the user.
- SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents, usually routers or servers. (see Fig.5.14)

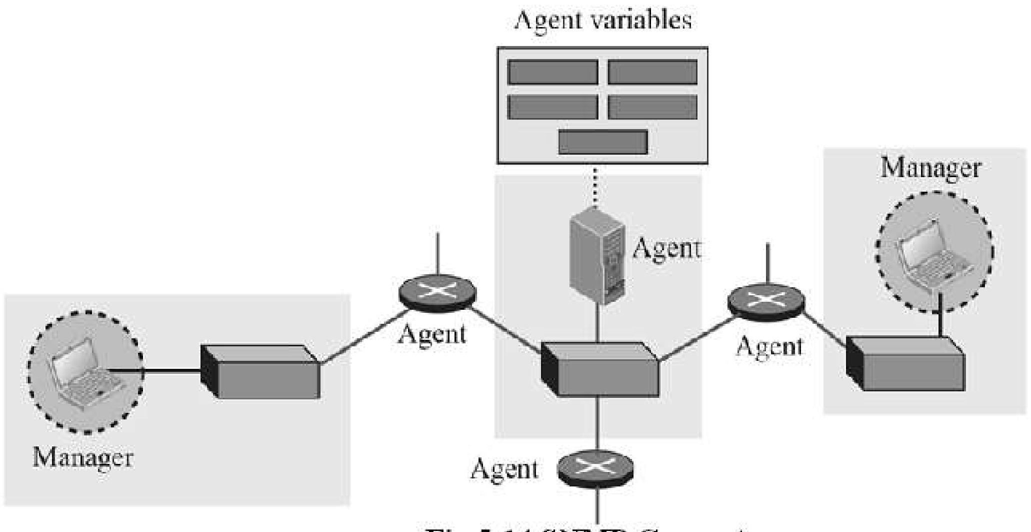


Fig 5.14 SNMP Concept

5.1.5. MANAGEMENT COMPONENTS

- To do management tasks, SNMP uses two other protocols: Structure of Management Information (SMI) and Management Information Base (MIB). In other words, management on the Internet is done through the cooperation of three protocols: SNMP, SMI, and MIB, as shown in Figure 5.15.
- **Role of SNMP**
  SNMP defines the format of packets exchanged between a manager and an agent. It reads and changes the status of objects (values of variables) in SNMP packets.

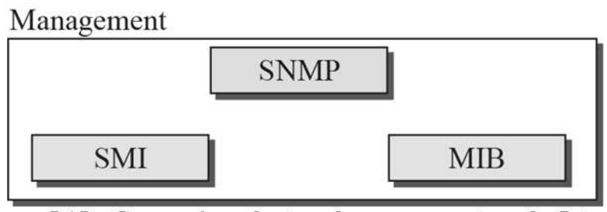


Figure 5.15 Companion of network management on the Internet

- **Role of SMI**
  SMI defines the general rules for naming objects, defining object types (including range and length), and showing how to encode objects and values.
- **Role of MIB**
  MIB creates a collection of named objects, their types, and their relationships to each other in an entity to be managed.
- SNMP depends on a companion specification called the *management information base* (MIB).

5.1.5.1. **SMI**

- The Structure of Management Information is a component for network management. Its functions are:

1. To name objects.
2. To define the type of data that can be stored in an object.
3. To show how to encode data for transmission over the network.

- SMI is a guideline for SNMP. It emphasizes three attributes to handle an object: *name, data type, and encoding method.*

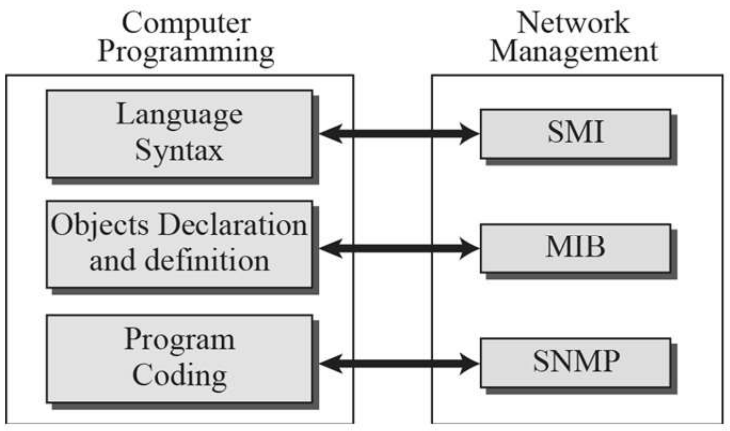


Figure 5.16 Comparing computer programming and network management

- ✓ ***Name*** *:* All objects managed by SNMP are given an object identifier. The object identifier always starts with 1.3.6.1.2.1. (see Fig 5.18)
- ✓ ***Encoding :*** Basic Encoding Rules (BER) defines a representation for different data types, such as integers.

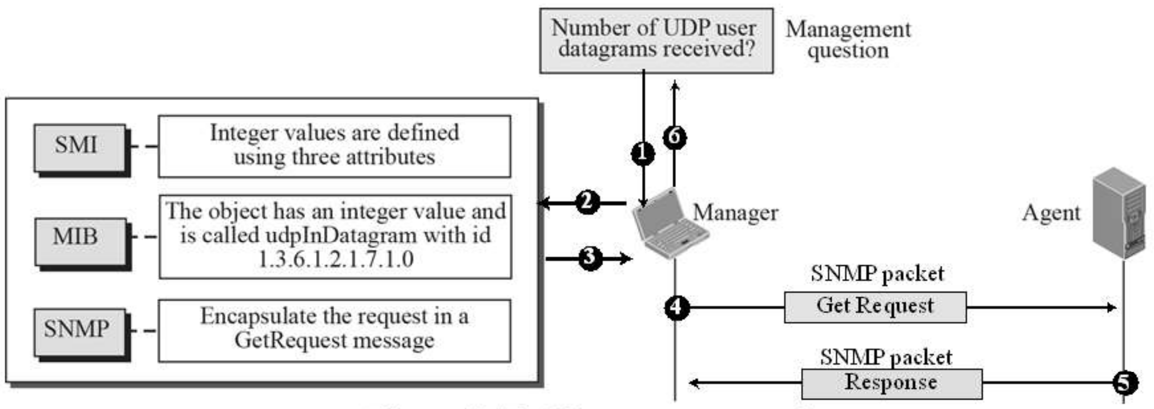

Number of UDP user datagrams received?
Management question
SMI
Integer values are defined using three attributes
MIB
The object has an integer value and is called udpInDatagram with id 1.3.6.1.2.1.7.1.0
SNMP
Encapsulate the request in a GetRequest message
1
2
3
4
5
6
Manager
Agent
SNMP packet
Get Request
SNMP packet
Response

Figure 5.17 ***Management overview***

Codes for Data Types

Data Type	*Tag (Binary)*	*Tag (Hex)*
INTEGER	00000010	02
OCTET STRING	00000100	04
OBJECT IDENTIFIER	00000110	06
NULL	00000101	05
Sequence, sequence of	00110000	30
IPAddress	01000000	40
Counter	01000001	41
Gauge	01000010	42
TimeTicks	01000011	43
Opaque	01000100	44

Data Types

Type	*Size*	*Description*
INTEGER	4 bytes	An integer with a value between -2^{31} and $2^{31}-1$
Integer32	4 bytes	Same as INTEGER
Unsigned32	4 bytes	Unsigned with a value between 0 and $2^{32}-1$
OCTET STRING	Variable	Byte-string up to 65,535 bytes long
OBJECT IDENTIFIER	Variable	An object identifier
IPAddress	4 bytes	An IP address made of four integers
Counter32	4 bytes	An integer whose value can be incremented from zero to 2^{32}; when it reaches its maximum value it wraps back to zero
Counter64	8 bytes	64-bit counter
Gauge32	4 bytes	Same as Counter32, but when it reaches its maximum value, it does not wrap; it remains there until it is reset
TimeTicks	4 bytes	A counting value that records time in 1/100ths of a second
BITS		A string of bits
Opaque	Variable	Uninterpreted string

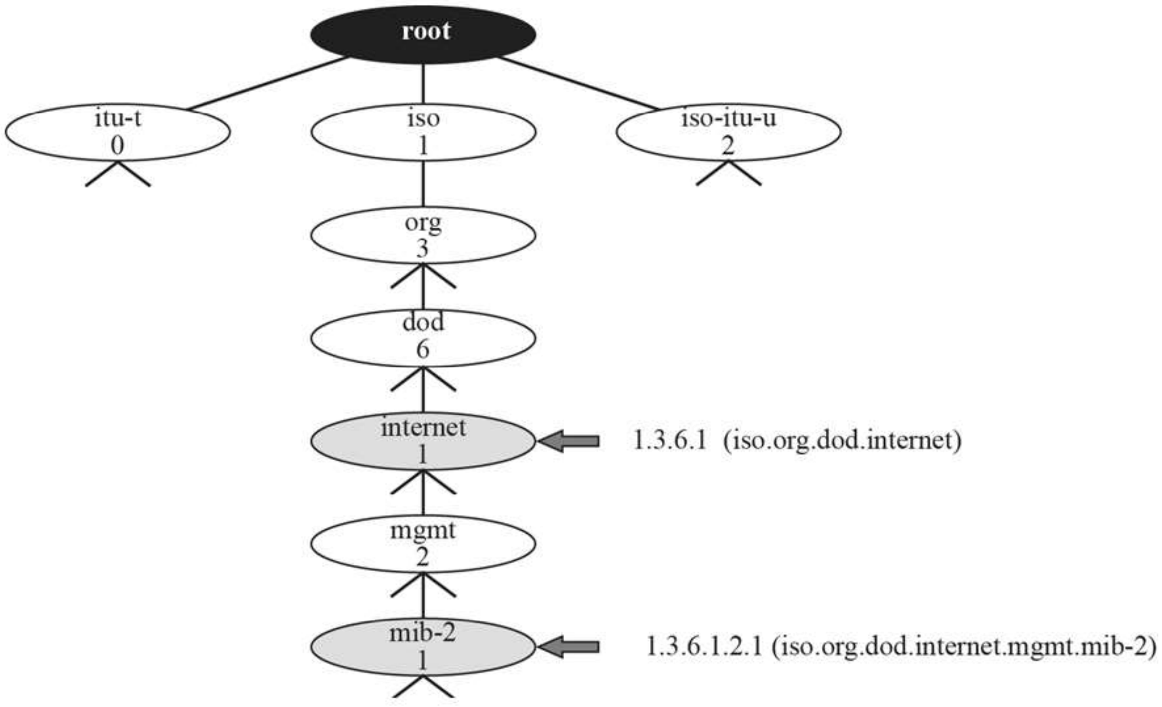

root
itu-t 0
iso 1
iso-itu-u 2
org 3
dod 6
internet 1
1.3.6.1 (iso.org.dod.internet)
mgmt 2
mib-2 1
1.3.6.1.2.1 (iso.org.dod.internet.mgmt.mib-2)

Figure 5.18 Object identifier

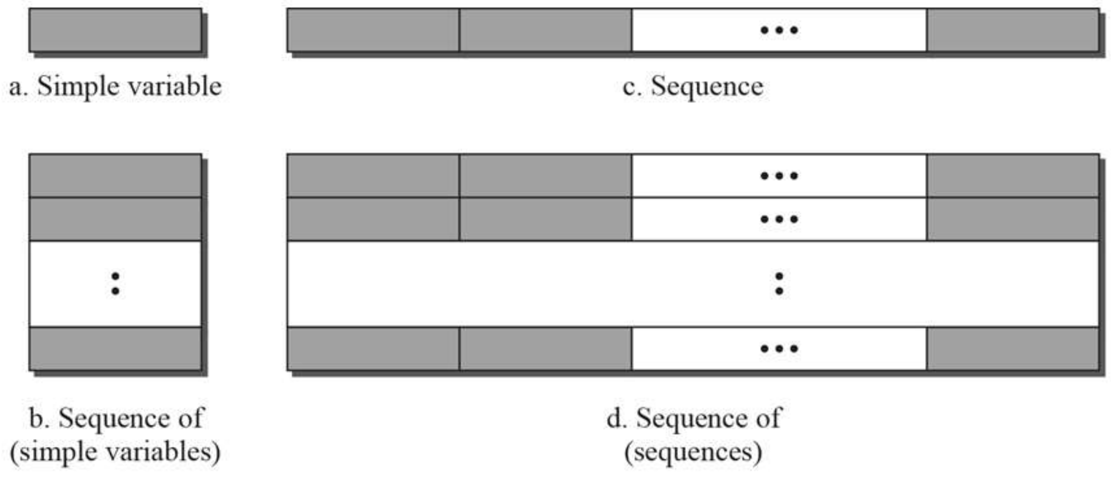


Figure 5.19 ***Conceptual data types***

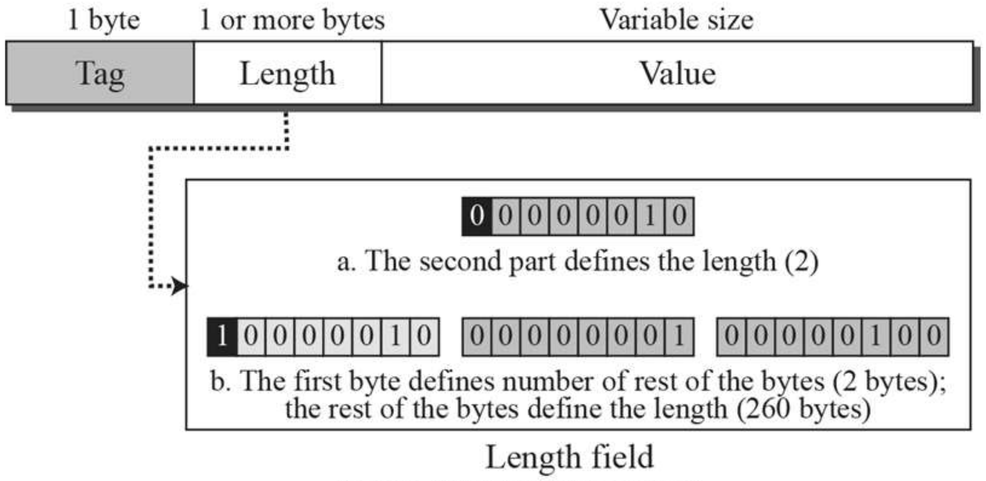

1 byte
1 or more bytes
Variable size
Tag
Length
Value
0 0 0 0 0 0 1 0
a. The second part defines the length (2)
1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0
b. The first byte defines number of rest of the bytes (2 bytes); the rest of the bytes define the length (260 bytes)
Length field

Figure 5.20 Encoding format

5.1.5.2. MIB

➢ The MIB defines the specific pieces of information—the MIB *variables*—that you can retrieve from a network node. (see Fig 5.21)

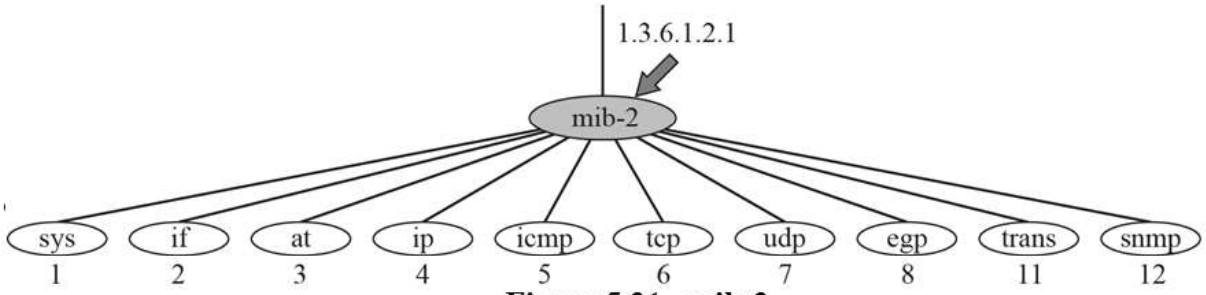


Figure 5.21 mib-2

❖ *System*—General parameters of the system (node) as a whole, including where the node is located, how long it has been up, and the system's name

❖ *Interfaces*—Information about all the network interfaces (adaptors) attached to this node, such as the physical address of each interface and how many packets have been sent and received on each interface

- ❖ *Address translation*—Information about the Address Resolution Protocol, and in particular, the contents of its address translation table
- ❖ *IP*—Variables related to IP, including its routing table, how many datagrams it has successfully forwarded, and statistics about datagram reassembly; includes counts of how many times IP drops a datagram for one reason or another
- ❖ *TCP*—Information about TCP connections, such as the number of
- ❖ passive and active opens, the number of resets, the number of timeouts, default timeout settings, and so on; per-connection information persists only as long as the connection exists
- ❖ *UDP*—Information about UDP traffic, including the total number of UDP datagrams that have been sent and received.

- ➢ There are also groups for Internet Control Message Protocol (ICMP), Exterior Gateway Protocol (EGP), and SNMP itself.
- ➢ The MIB defines the type of each variable, and then it uses ASN.1/BER to encode the value contained in this variable as it is transmitted over the network.
- ➢ The MIB uses this identification system to assign a globally unique identifier to each MIB variable.
- ➢ SNMP client puts the ASN.1 identifier for the MIB variable it wants to get into the request message, and it sends this message to the server.

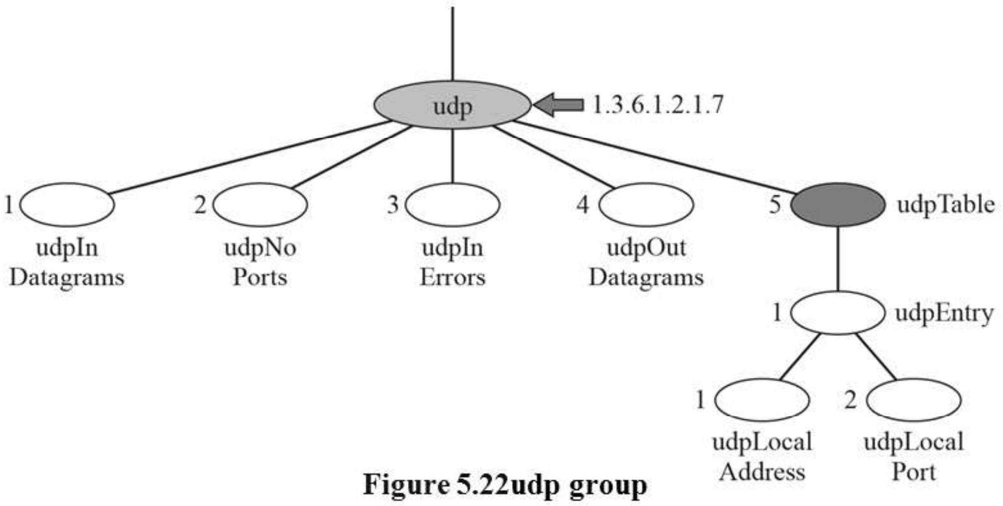


Figure 5.22udp group

- ➢ The server then maps this identifier into a local variable retrieves the current value held in this variable, and uses ASN.1/BER to encode the value it sends back to the client.
- ➢ Many of the MIB variables are either tables or structures. Such compound variables explain the reason for the SNMP GET-NEXT operation

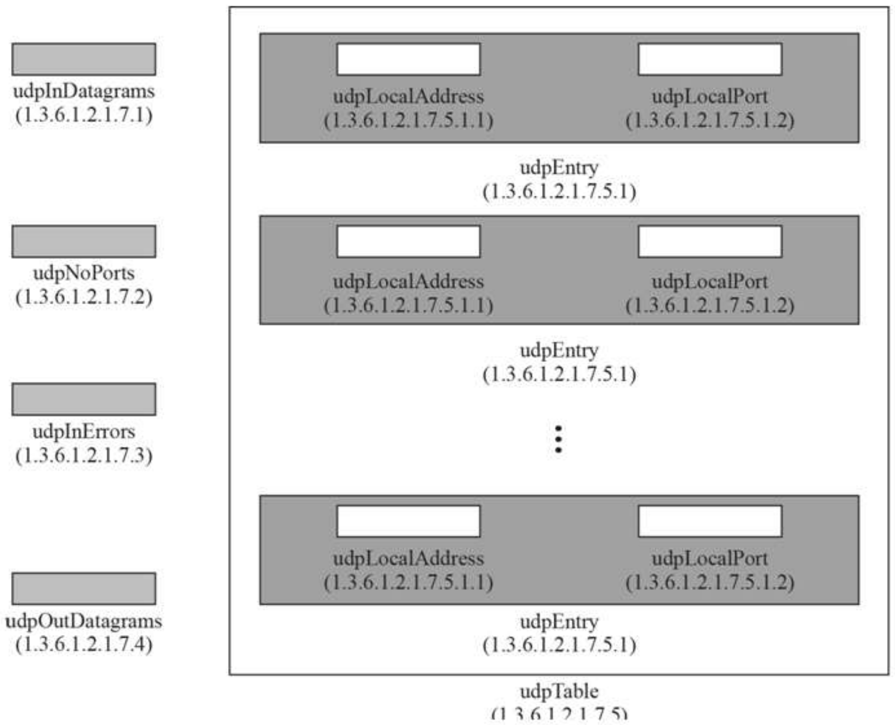


Figure 5.23 udp variables and tables

- . This operation, when applied to a particular variable ID, returns the value of that variable plus the ID of the next variable, for example, the next item in the table or the next field in the structure. This aids the client in "walking through" the elements of a table or structure.

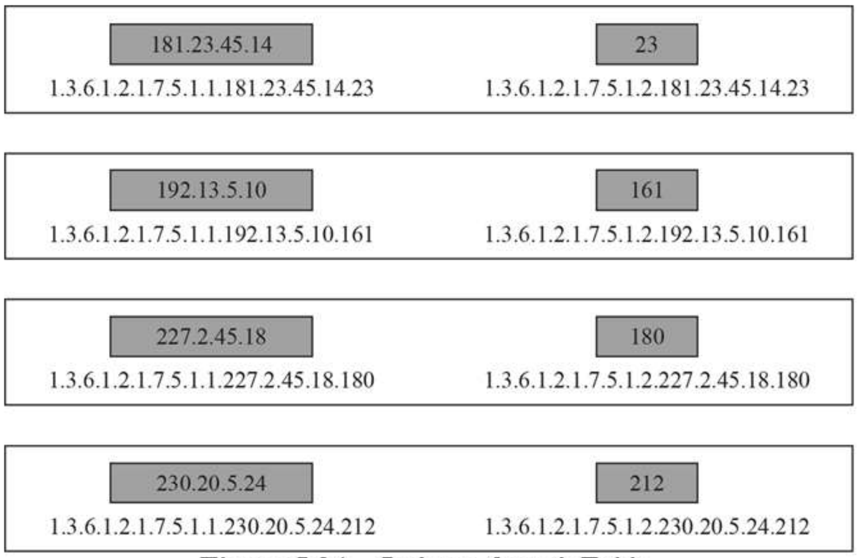


Figure 5.24 ***Indexes for udpTable***

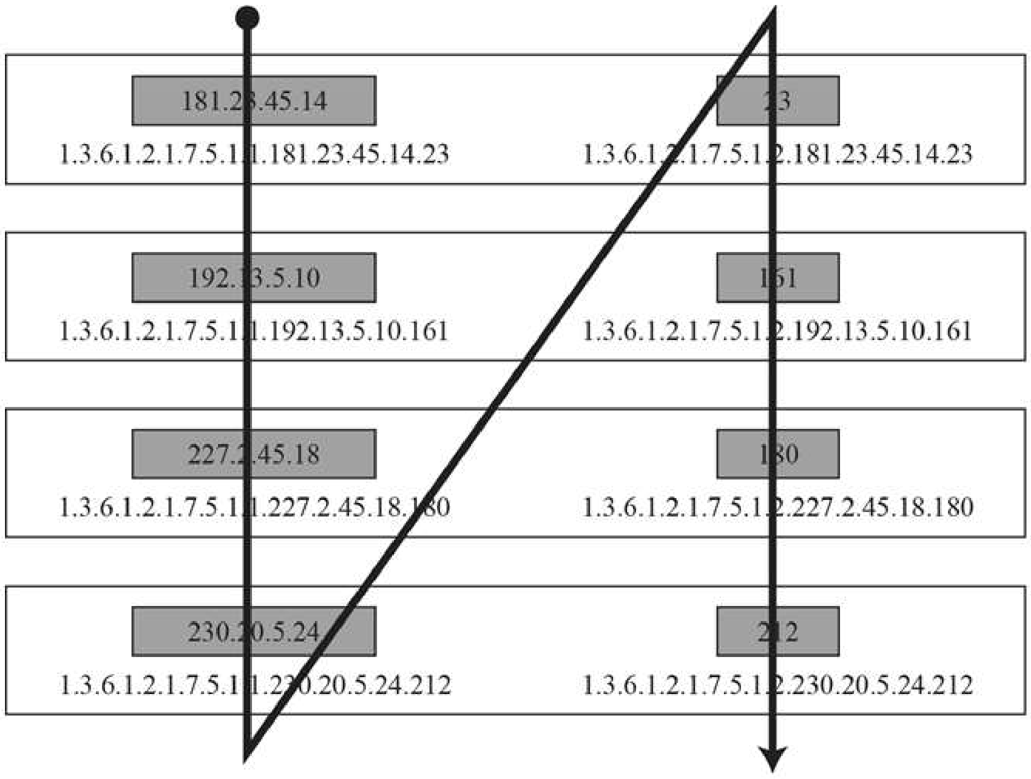


Figure 2.25 ***Lexicographic ordering***

➢ SNMP is an application program that allows:
1. A manager to retrieve the value of an object defined in an agent.
2. A manager to store a value in an object defined in an agent.
3. An agent to send an alarm message about an abnormal situation to the manager.

✓ ***Protocol Data Unit***

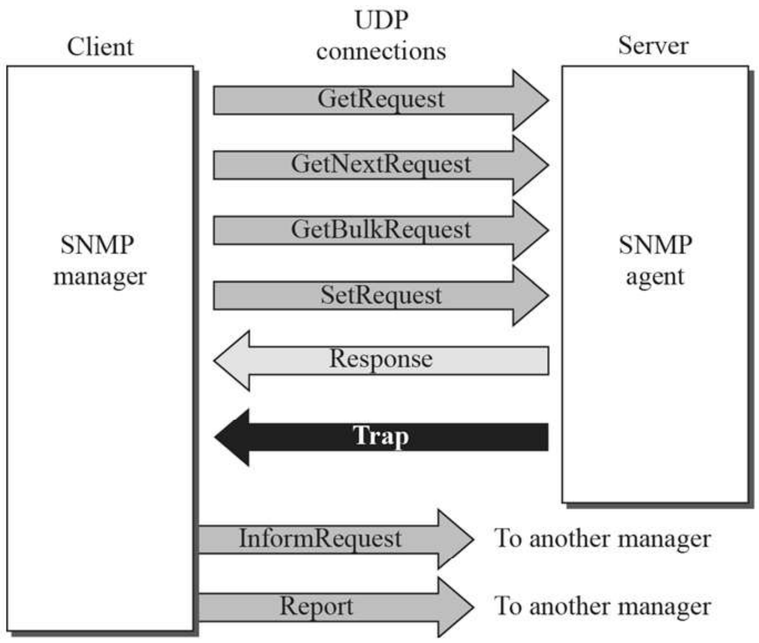
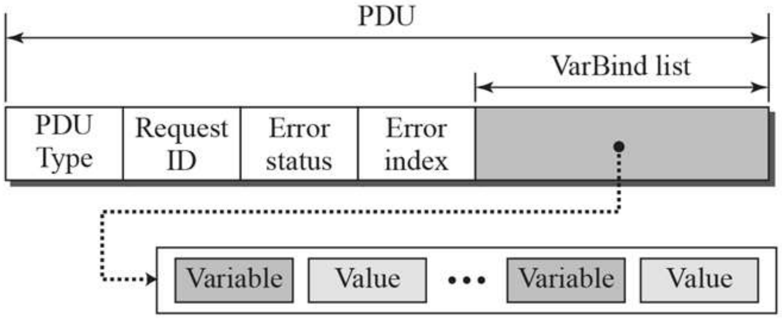


Figure 5.26 SNMP PDUs

✓ ***Format***

PDU Types

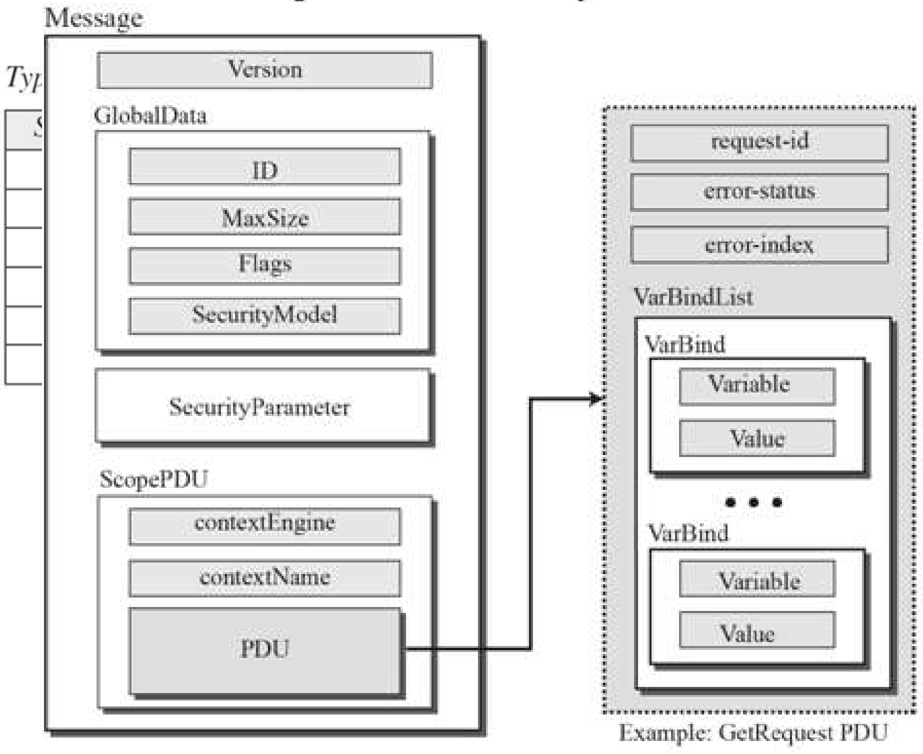
Type	*Tag (Binary)*	*Tag (Hex)*
GetRequest	**10100000**	**A0**
GetNextRequest	**10100001**	**A1**
Response	**10100010**	**A2**
SetRequest	**10100011**	**A3**
GetBulkRequest	**10100101**	**A5**
InformRequest	**10100110**	**A6**
Trap (SNMPv2)	**10100111**	**A7**
Report	**10101000**	**A8**



Differences:

1. Error status and error index values are zeros for all request messages except GetBulkRequest.
2. Error status field is replaced by non-repeater field and error index field is replaced by max-repetitions field in GetBulkRequest.

Figure 5.27 ***SNMP PDU format***



✓ **Messages**

Figure 5.28 ***SNMP message***

5.1.5.3. **UDP PORTS**

✓ SNMP uses the services of UDP on two well-known ports, 161 and 162. The well-known port 161 is used by the server (agent), and the well-known port 162 is used by the client (manager).

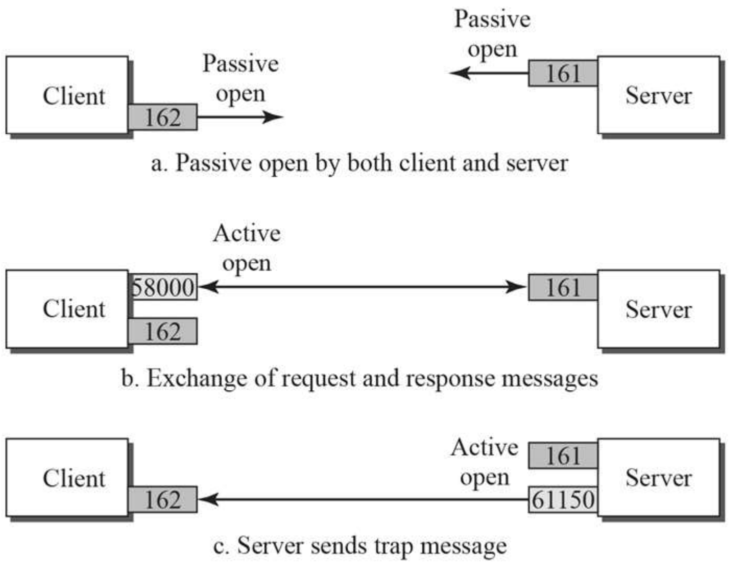

Passive open
Client
162
Passive open
161
Server
a. Passive open by both client and server
Active open
Client
58000
162
161
Server
b. Exchange of request and response messages
Client
162
Active open
161
61150
Server
c. Server sends trap message

Figure 5.29 ***Port numbers for SNMP***