Module 7
Application Layer

BECE401L

WWW

World Wide Web

Idea of the Web was first proposed by Tim Berners-Lee in 1989 at CERN, EU-NR.

Commercial Web started in the early 1990s.

Web today is a repository of information, called ***web pages***.

- Documents are distributed all over the world and related ones are linked together

Web can be related to two terms: Distributed & Linked.

Distribution

- Allows the growth of the Web.
- Each web server in the world can add a new web page to the repository and announce it to all Internet users without overloading a few servers.

Linking

- Allows one web page to refer to another web page stored in another server somewhere else in the world.
- Linking of web pages was achieved using a concept called ***hypertext***,

WWW: Architecture

Distributed Client-Server Service:

- In which a client using a browser can **access a service using a server**.
- Service provided is distributed over many locations called ***sites***.
- Each site holds one or more web pages.

Web page:

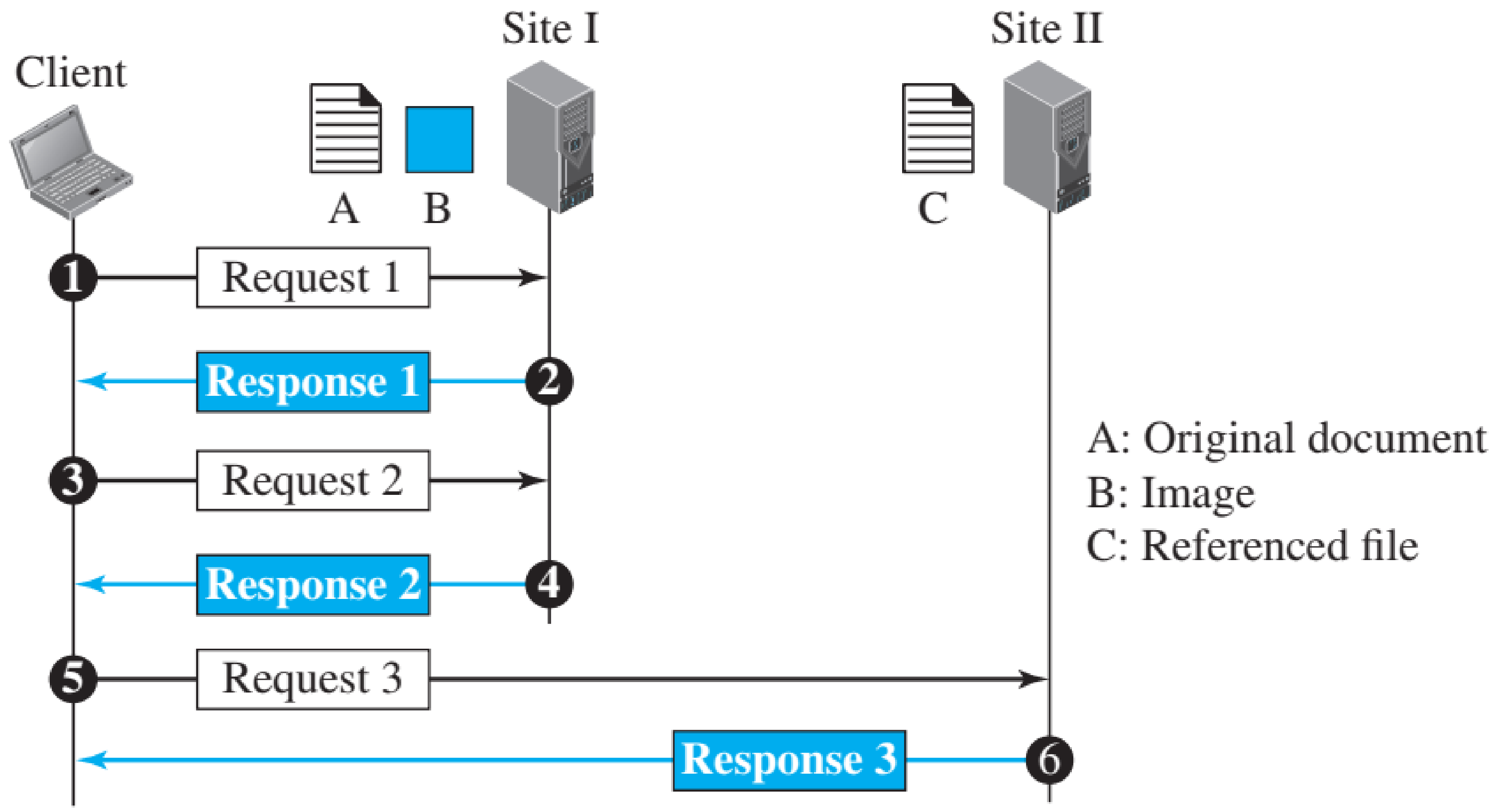
- Can contain some links to other web pages in the same or other sites.
- No links to other web pages; a composite web page.
- **Each web page is a file with a name and address**.

WWW: Architecture Example

Assume we need to retrieve a scientific document that contains one reference to another text file and one reference to a large image.

Example:

The main document and the image are stored in two separate files (file A and file B) in the same site; the referenced text file (file C) is stored in another site.

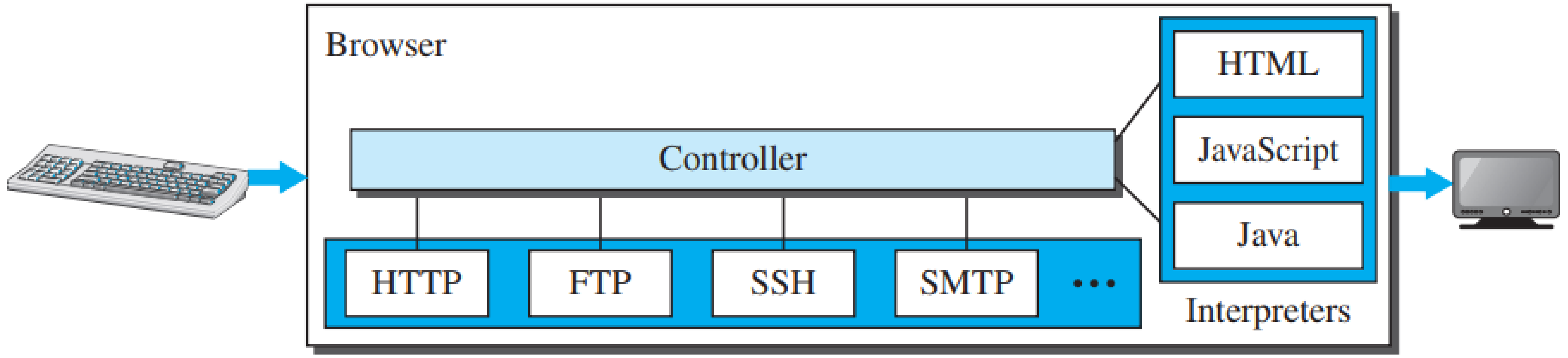


WWW: Web Client

Browsers: interpret and display a web page.

Consists of three parts:

- **Controller:**
  - receives input from the keyboard or the mouse and uses the client programs to access the document
- **Client protocols:**
  - The client protocol can be one of the protocols, such as HTTP or FTP.
- **Interpreters:**
  - Can be HTML, Java, or JavaScript, depending on the type of document.



WWW: Web Server

Web page is stored at the Server.

Each time a request arrives: Corresponding document is sent to the client.

Servers normally store requested files in a cache in memory.

Multithreading or Multiprocessing:

- Can improve efficiency.

WWW: URL

Uniform Resource Locator (URL)

To define a web page, we need three identifiers: **host**, **port**, and **path**.

Before defining the web page, we need to tell the browser what client-server application we want to use, which is called **the protocol**.

Protocol.

- The first identifier is the abbreviation for the client-server program that we need in order to access the web page.

Host.

- The host identifier can be the IP address of the server or the unique name given to the server.

Port.

- The port, a 16-bit integer, is normally predefined for the client-server application. Eg. for HTTP port no. is 80

Path.

- The path identifies the location and the name of the file in the underlying operating system.

protocol://host/path	Used most of the time
protocol://host:port/path	Used when port number is needed

WWW: Documents

3 Broad Categories: Static, Dynamic and Active

Static Documents.

- **Fixed-content** documents that are created and stored in a server.
- The client can get a copy of the document only.
  - When a client accesses the document, a copy of the document is sent and the user can then use a browser to see the document.
- Static documents are prepared using one of several languages: HTML, XML, XSL and XHTML.

Dynamic Documents

- Created by a web server whenever a browser requests the document.
  - When a request arrives, the web server runs an application program or a script that creates the dynamic document.
  - The server returns the result of the program or script as a response to the browser that requested the document.
- **Example:** Retrieval of the time and date from a server.
- Scripting languages such as *Java Server Pages* (*JSP*), or *Active Server Pages* (*ASP*).

Active Documents

- Program or a script to be run at the client site.
- **Example:** A program that creates animated graphics or a program that interacts with the user.
- **Scripting Language:** *Java Applets*

HTTP

HyperText Transfer Protocol (HTTP)

- Define how the client-server programs can be written to retrieve web pages from the Web.

Port Nos.

- Server uses the port number **80**.
- Client uses a temporary port number

HTTP uses the services of TCP, is a connection-oriented and reliable protocol.

- Before any transaction between the client and the server can take place,
- A connection needs to be established between them.
- After the transaction, the connection should be terminated.

HTTP: Connections

Nonpersistent Connections

One TCP connection is made for each request/response.

Steps

- The client opens a TCP connection and sends a request.
- The server sends the response and closes the connection.

Client reads the data until it encounters an end-of-file marker;

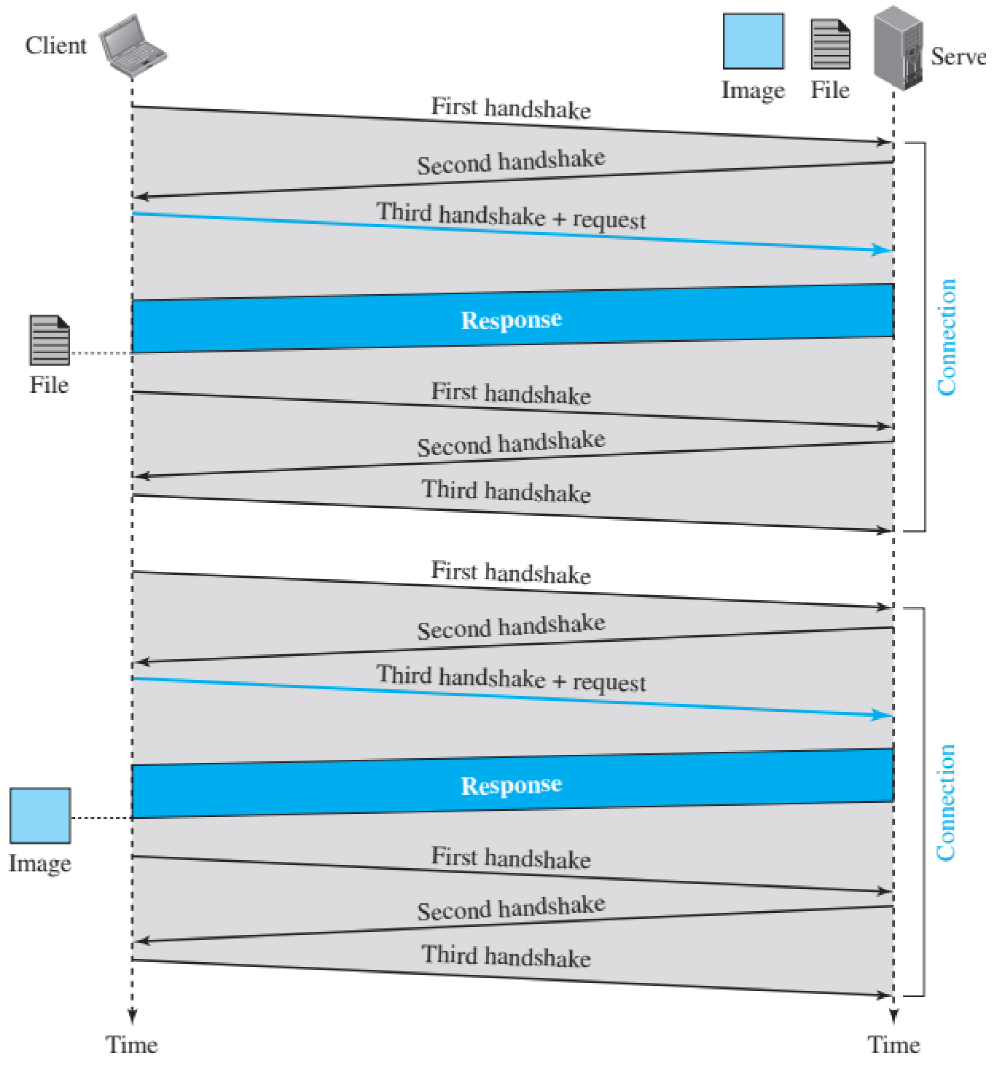
- It then closes the connection.

In this strategy, if a **file contains links to *N* different pictures in different files**

- The connection must be opened and closed *N* + 1 times.

Imposes high overhead on the server

- Because the server needs *N* + 1 different buffers each time a connection is opened



HTTP: Connections

Persistent Connections

HTTP version 1.1 specifies a **persistent connection** by default.

In a persistent connection,. the server leaves the connection open for more requests after sending a response.

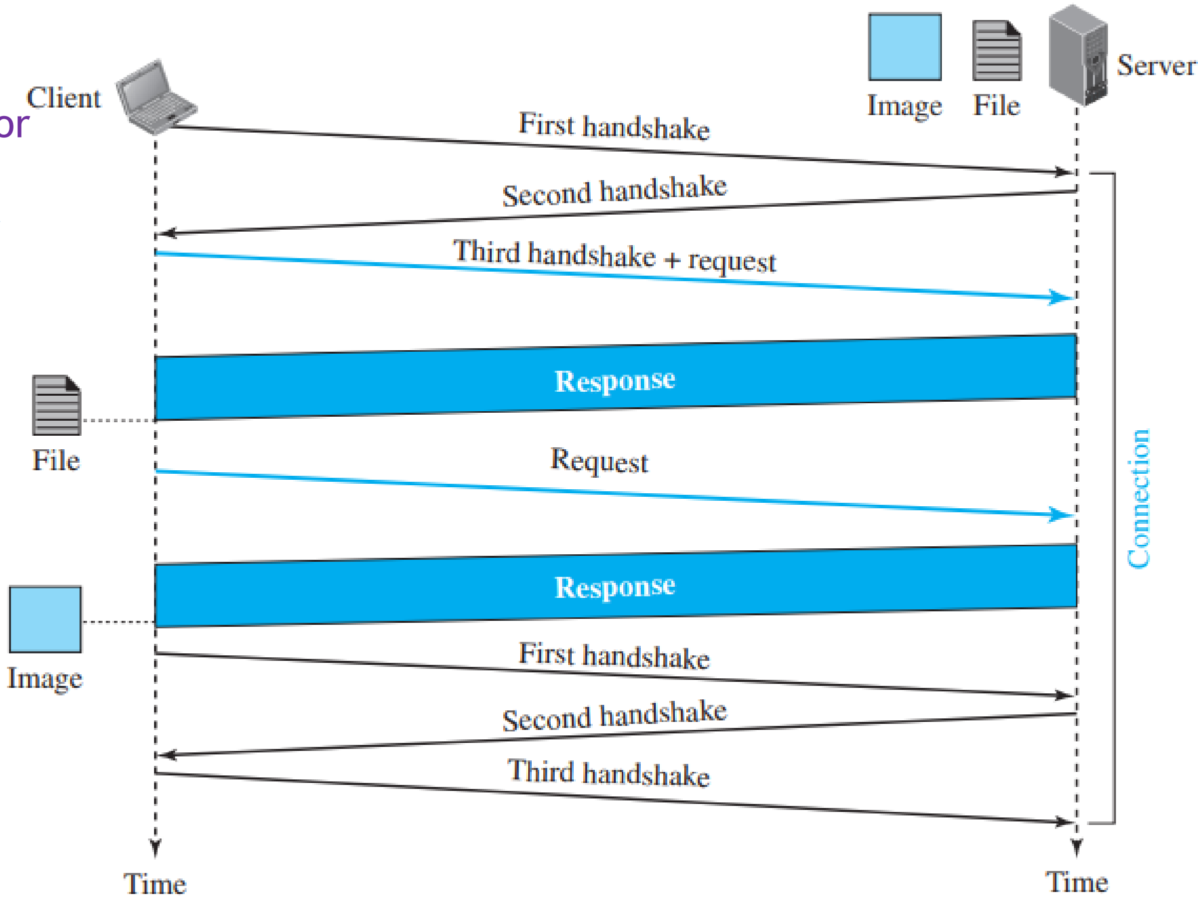
Server can close the connection at the request of a client or at time-out.

- Sender usually sends the length of the data with each response.
- There are some occasions when the sender does not know the length of the data.
- This is the case when a document is created dynamically or actively.
  - In these cases, the server informs the client that the length is not known and closes the connection after sending the data.
  - So the client knows that the end of the data has been reached.

Time and resources are saved using persistent connections.

Only one set of buffers and variables needs to be set for the connection at each site.

The round trip time for connection establishment and connection termination is saved

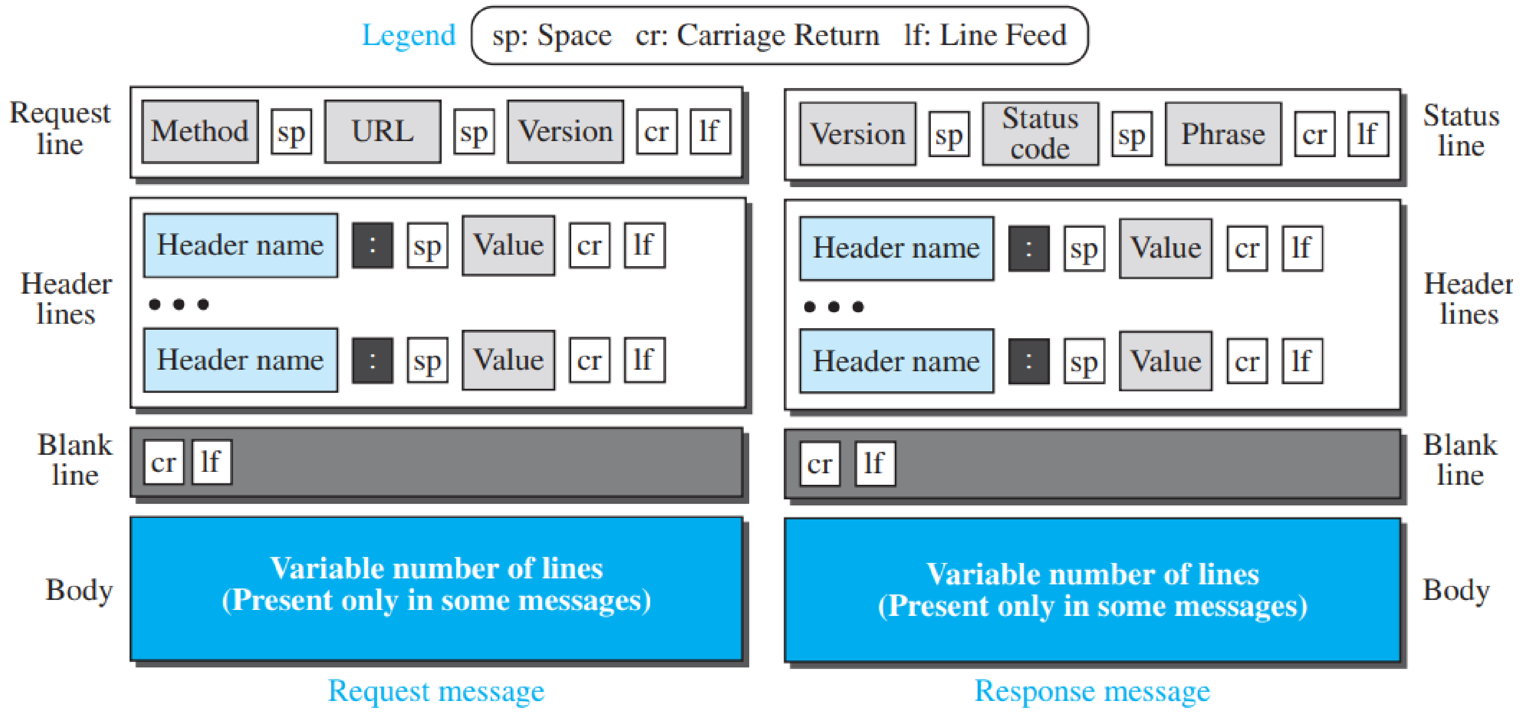


HTTP: Message Format

The HTTP protocol defines the format of the request and response messages

Each message is made of **four** sections.

- The first section: In the request message is called the *request line*; In the response message is called the *status line*.
- The other three sections have the same names in the request and response messages.



Legend
sp: Space cr: Carriage Return lf: Line Feed
Request line
Method sp URL sp Version cr lf
Header lines
Header name : sp Value cr lf
• • •
Header name : sp Value cr lf
Blank line
cr lf
Body
Variable number of lines (Present only in some messages)
Request message
Status line
Version sp Status code sp Phrase cr lf
Header lines
Header name : sp Value cr lf
• • •
Header name : sp Value cr lf
Blank line
cr lf
Body
Variable number of lines (Present only in some messages)
Response message

HTTP: Message Format

Methods:

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

Request Header Names

Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

DNS

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet.

Not feasible for humans to remember millions of IP Addresses.

So, Internet needs to have a directory system that can map a name to an address.

As, the Internet is so huge today, a central directory system cannot hold all the mapping.

In addition, if the central computer fails, **the whole communication network will collapse.**

A better solution is to **distribute the information** among many computers in the world.

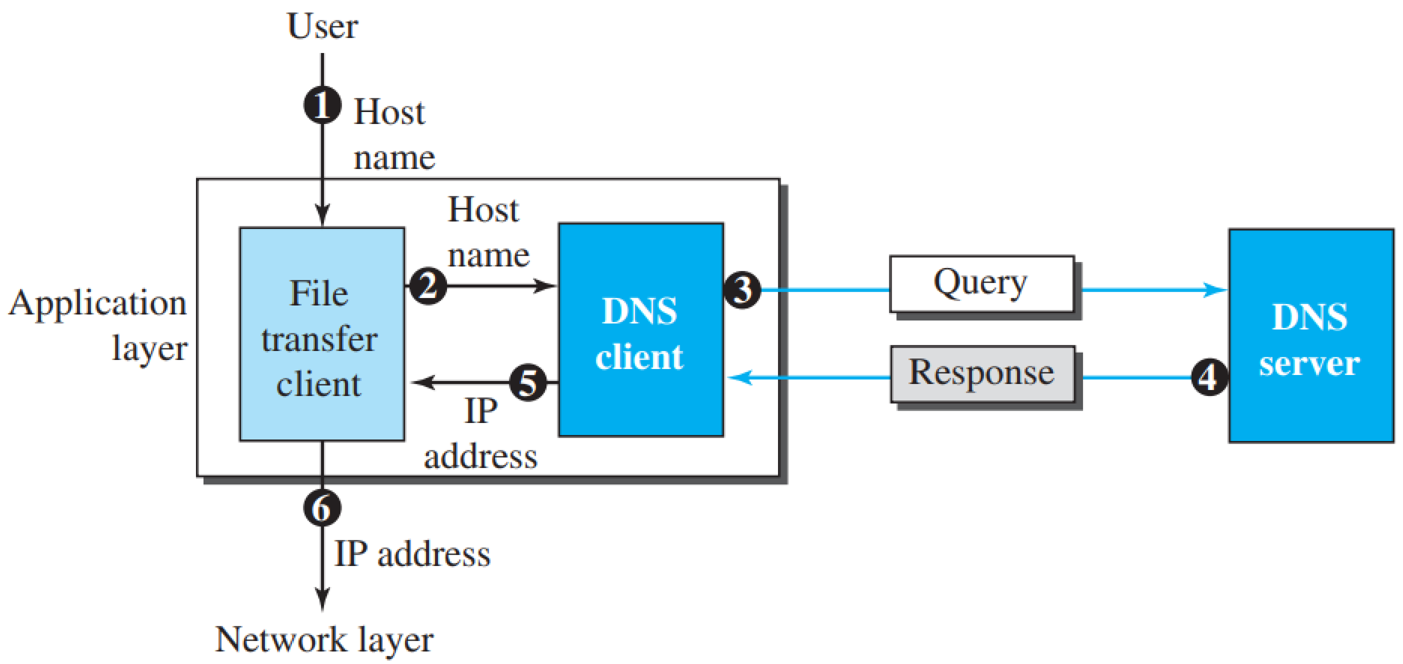
- In this method, the **host that needs mapping can contact the closest computer holding the needed information**.

This method is used by the Domain Name System (DNS).

Purpose of DNS

Figure shows how TCP/IP uses a DNS client and a DNS server to map a name to an address.

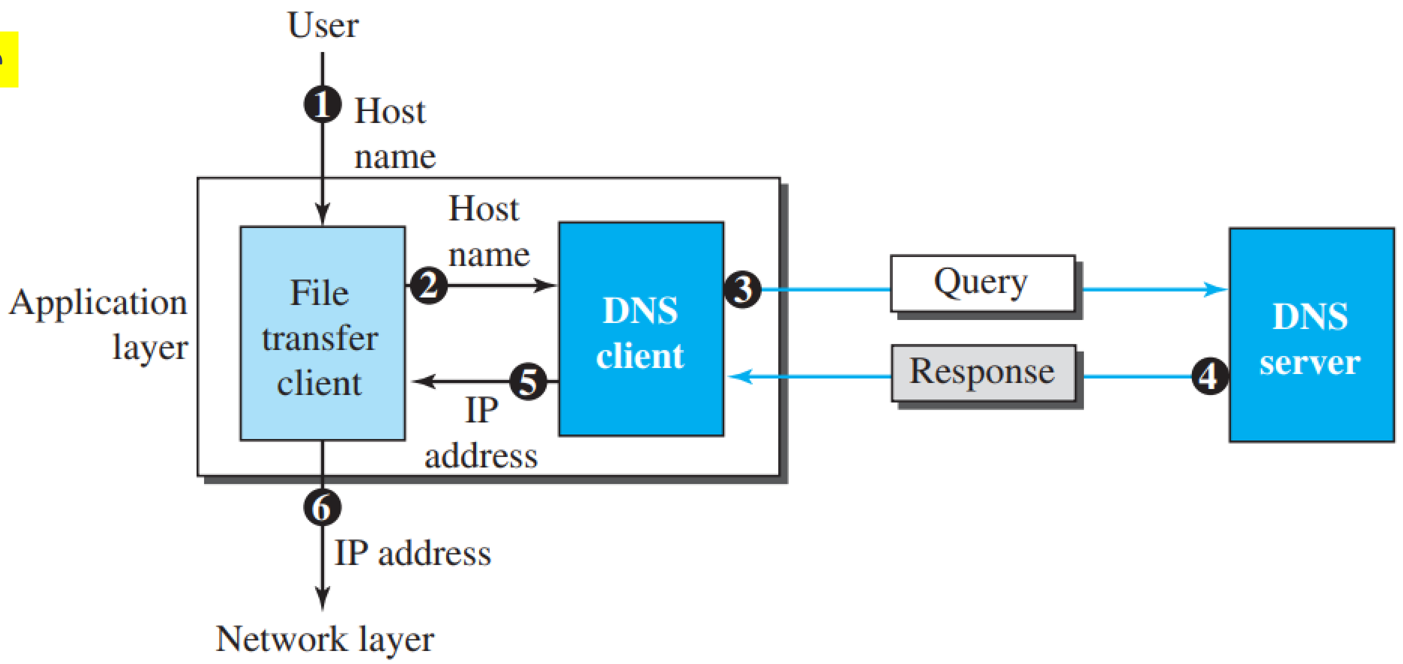
- A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host.
- The user knows only the server name, such as afilesource.com.
- However, the TCP/IP suite needs the IP address of the file transfer server to make the connection.



Purpose of DNS

6 step procedure is executed to map the host name to an IP address:

1. The user passes the host name to the file transfer client.
2. The file transfer client passes the host name to the DNS client.
3. Each computer, after being booted**, knows the address of one DNS server**.
   - DNS client **sends a message to a DNS server with a query** that **gives the file transfer server name using the known IP address of the DNS server.**
4. DNS server responds with the IP address of the desired file transfer server.
5. DNS server passes the IP address to the file transfer client.
6. The file transfer client now uses the received IP address to access the file transfer server.



DNS

Name Space:

- The names must be unique because the addresses are unique.
- A name space that maps each address to a unique name can be organized in two ways:
  - Flat
  - Hierarchical.

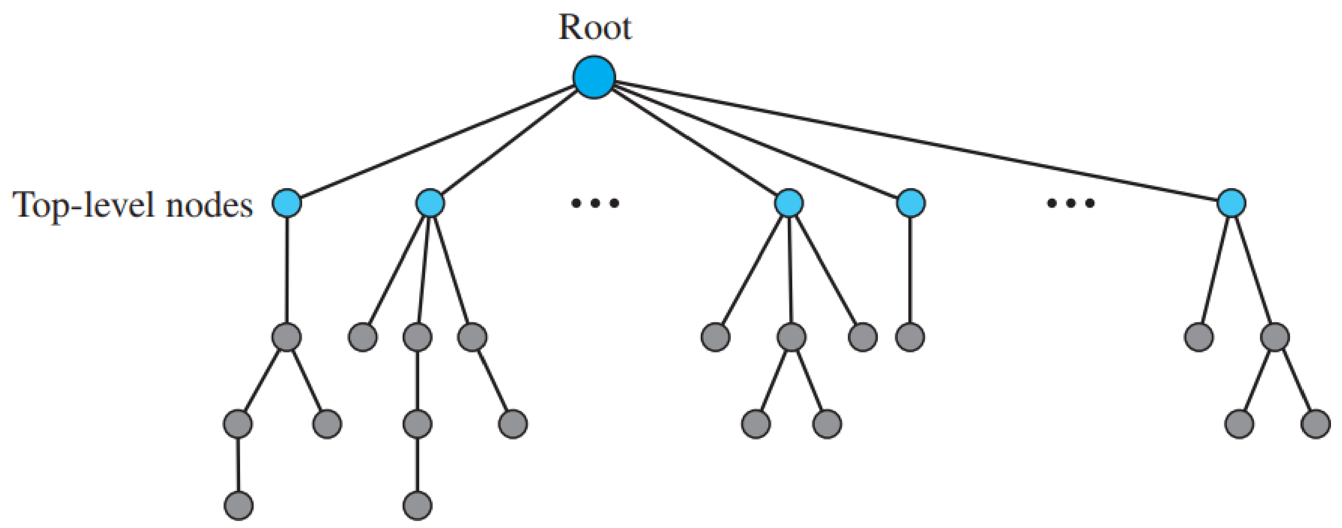
- **Flat name space,**
  - A name is assigned to an address.
  - A name in this space is a ***sequence of characters without structure***.
  - The names may or may not have a common section; if they do, it has no meaning.
  - The main disadvantage of a flat name space is that it ***cannot be used in a large system such as the Internet*** because it must be centrally controlled to avoid ambiguity and duplication.

- **Hierarchical name space,**
  - Each name is made of several parts.
  - First part can define the nature of the organization,
  - Second part can define the name of an organization,
  - Third part can define departments in the organization, and so on.
  - In this case, the authority to assign and control the name spaces can be decentralized.

DNS

Domain Name Space:

- To have a hierarchical name space, a domain name space was designed.
- In this design the names are defined in an inverted-tree structure with the root at the top.
- The tree can have only 128 levels: level 0 (root) to level 127



DNS

Label:

- Each **node in the tree has a label**, which is a string with a **maximum of 63 characters**.
- The **root label is a null string** (empty string).
- DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

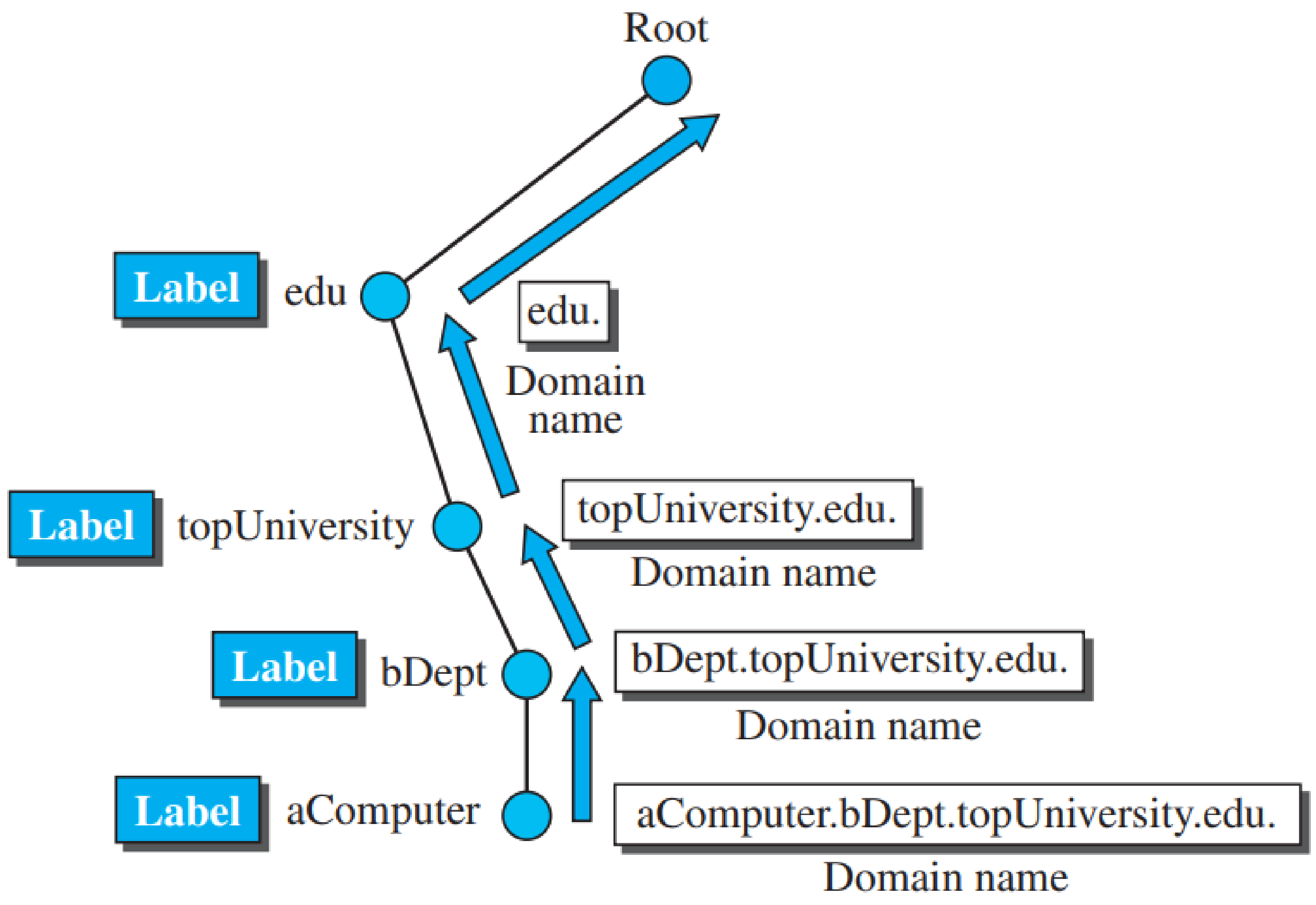
Domain Name:

- Each node in the tree has a domain name.
- A **full domain name is a sequence of labels separated by dots (.).**
- The domain names are always **read from the node up to the root**.
- The last label is the label of the root (null).
- This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing.

DNS

Label:

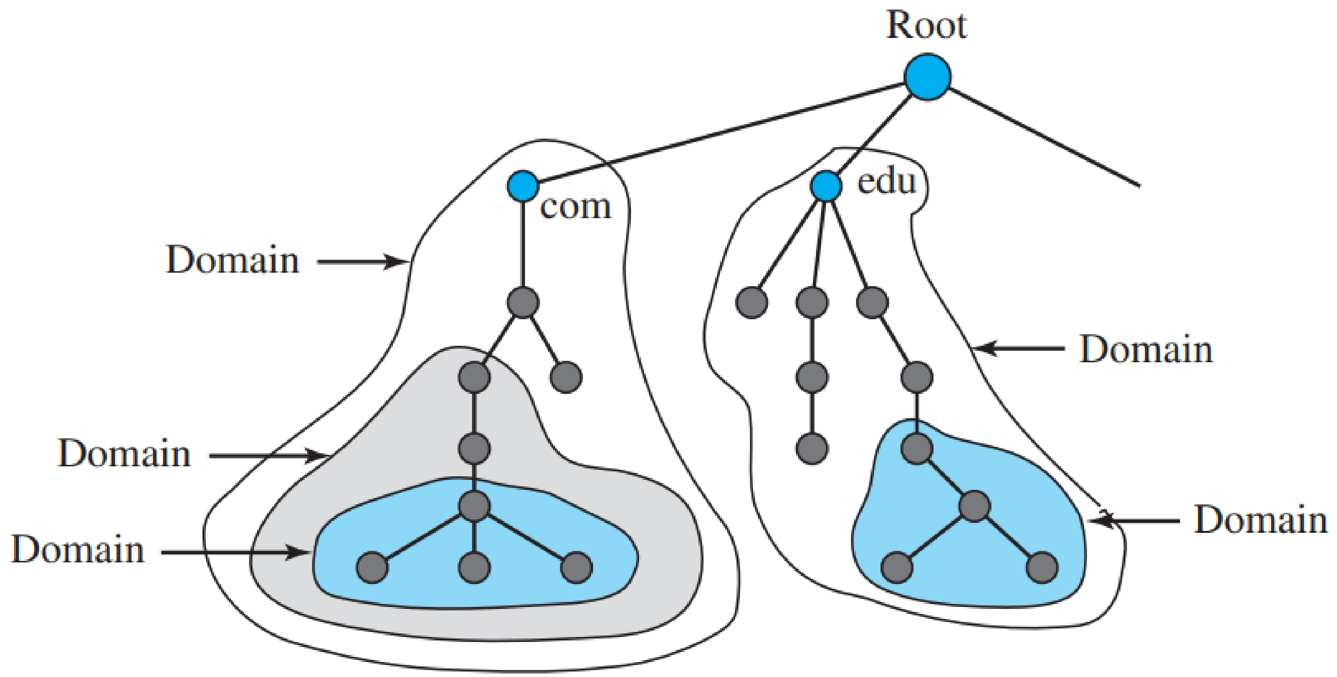
- Figure shows some domain names.



DNS

Domains:

- A domain is a subtree of the domain name space.
- The name of the **domain is the name of the node at the top of the subtree.**
- Note that a domain may itself be divided into sub - domains.

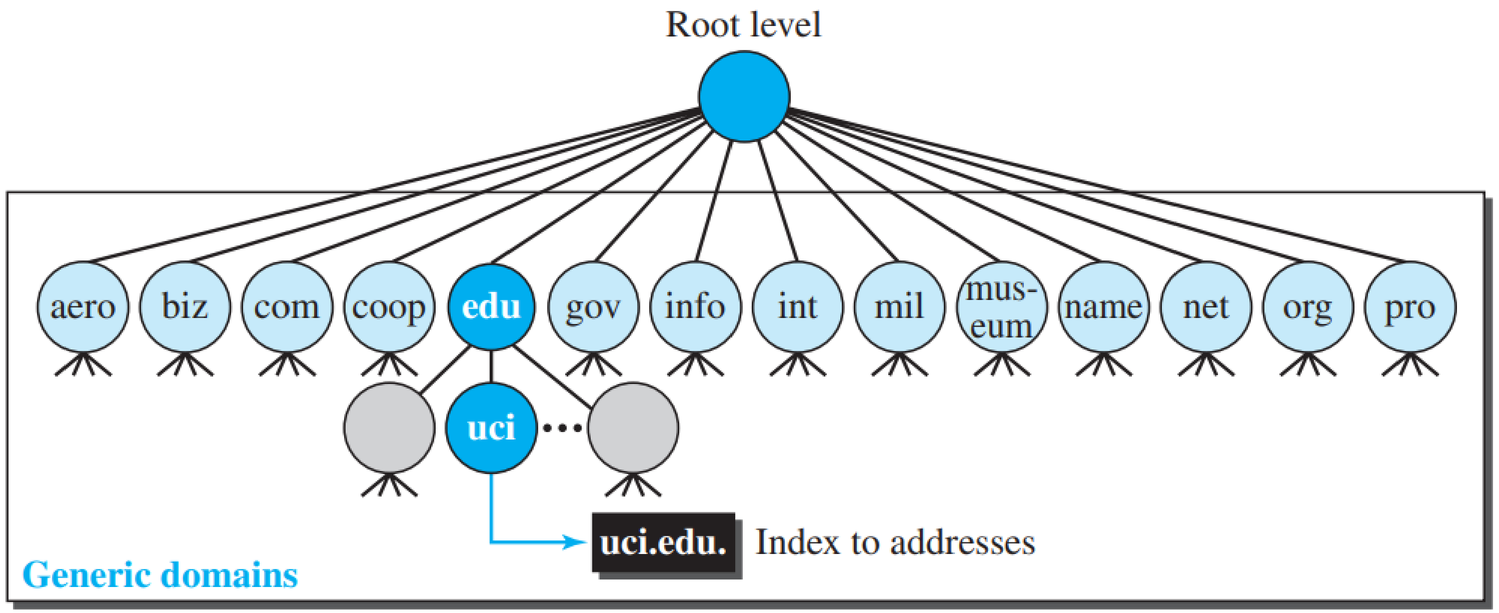


DNS in Internet

- In the Internet, the domain name space (tree) was originally divided into three different sections: generic domains, country domains, and the inverse domains.

Generic Domains:

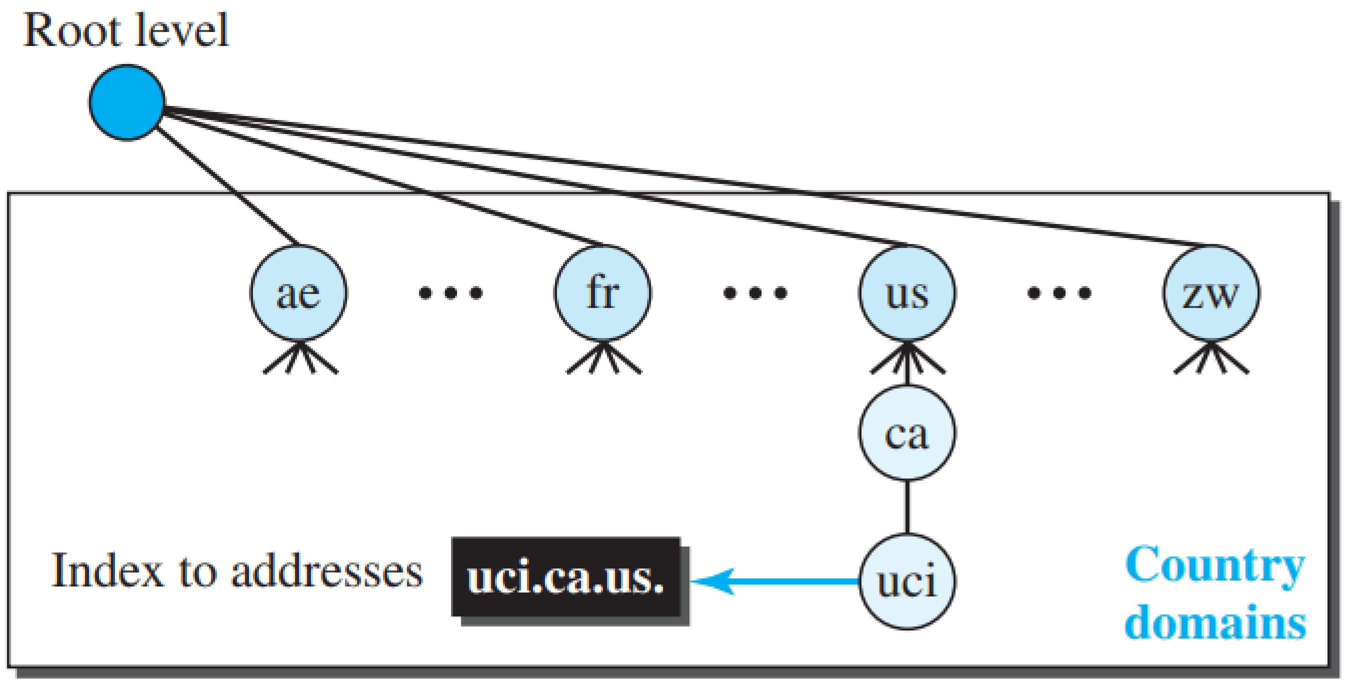
- The generic domains define registered hosts according to their generic behavior.
- Each node in the tree defines a domain, which is an index to the domain name space database



DNS in Internet

Country Domains:

- Country domains section uses two-character country abbreviations (e.g., us for United States).
- Second labels can be organizational, or they can be more specific national designations.



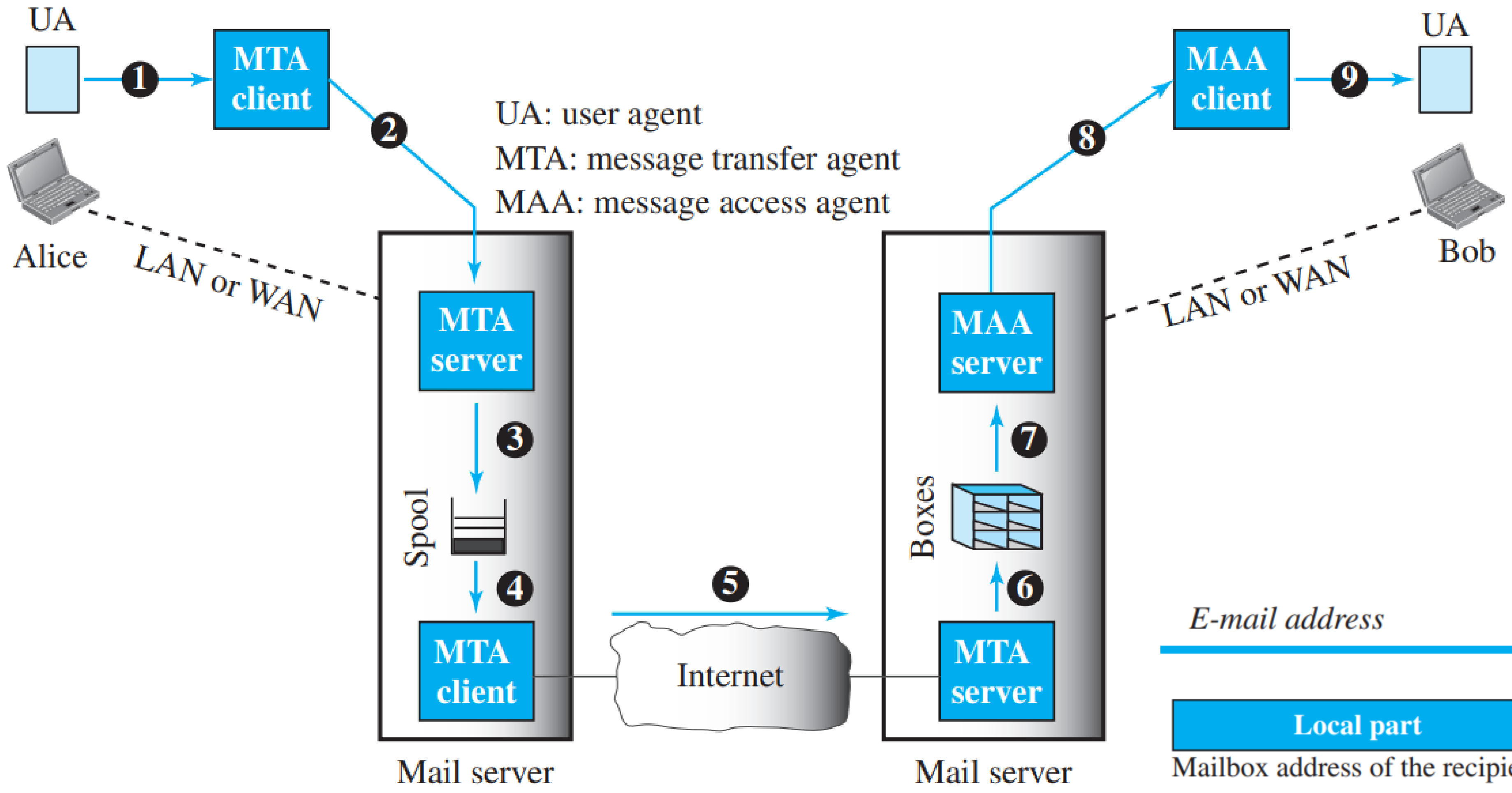
Email

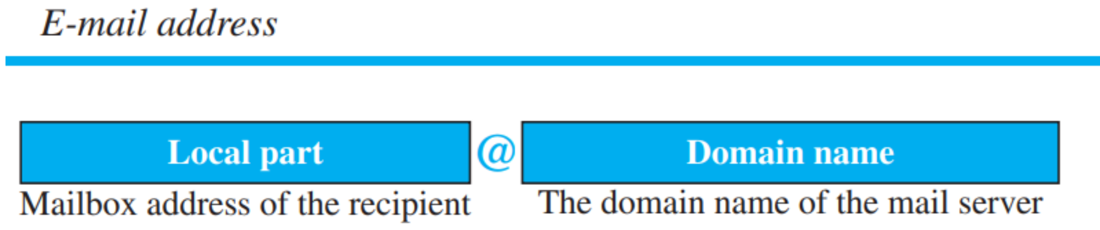
- Electronic mail (or e-mail) allows users to exchange messages.
- Nature of Email, however, is different from other applications such as HTTP or FTP.
    - In HTTP or FTP the **server program is running all the time**,
        - Waiting for a request from a client.
        - When the request arrives, the server provides the service.
        - There is a request and there is a response.
- In the case of EMail, the situation is different:
- First, e-mail is considered a one-way transaction.
    - **Example:** When Alice sends an email to Bob, she may expect a response, but this is not a mandate.
        - Bob may or may not respond.
        - If he does respond, it is another one-way transaction.

Email

- Second, it **is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him.**
  - Bob may turn off his computer when he is not using it.
  - This means that the **idea of client/server programming should be implemented in another way**: **using some intermediate computers (servers).**
  - **The users run only client programs when they want and the intermediate servers apply the client/server paradigm.**

Email : Architecture





E-mail address
Local part
@
Domain name
Mailbox address of the recipient
The domain name of the mail server

Email : Architecture

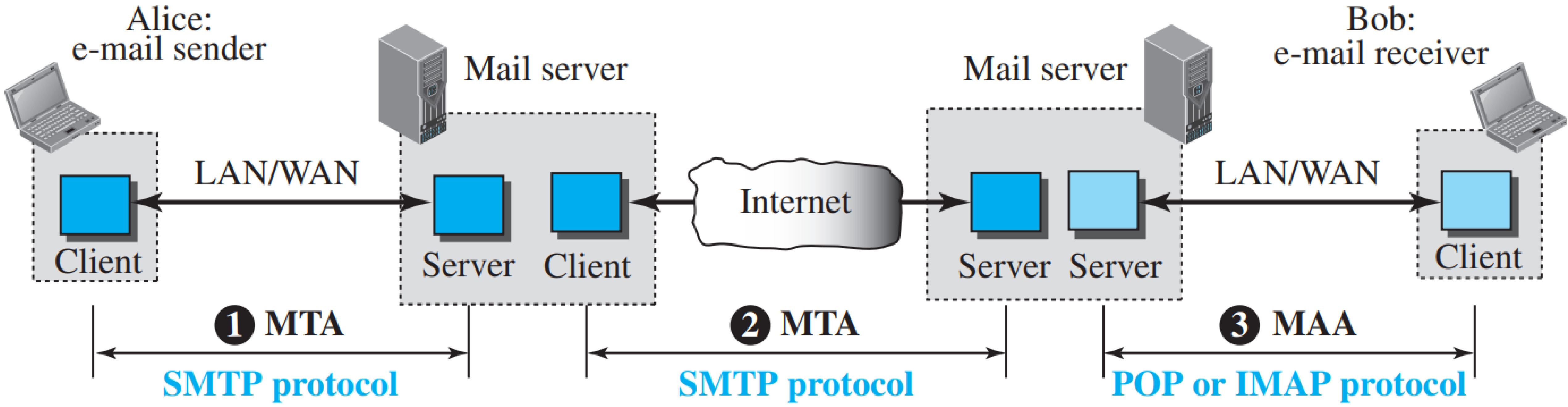
- A simple e-mail from Alice to Bob takes **nine different steps**, as shown in the figure.
- Alice and Bob use three different agents:
  - A user agent (UA),
  - A message transfer agent (MTA), and
  - A message access agent (MAA).
- **When Alice needs to send a message to Bob**,
  - She runs a **UA program** to prepare the message and send it to her mail server.
  - The **mail server at her site uses a queue (spool)** to store messages waiting to be sent.
  - The message, however, needs to be sent through the Internet from Alice's site to Bob's site using an **MTA**.
    - *Here two message MTA transfer agents are needed:*
      - Client : Triggered by the system when there is a message in queue to be sent
      - Server : Server needs to run all the time to ensure availability

Email : Architecture

- UA at the Bob site
  - Allows Bob to read the received message.
  - Bob later uses an MAA client to ***retrieve the message*** from an MAA server ***running on the second server***
- **Important Points:**
  - Bob cannot bypass the **mail server** and use the MTA server directly.
    - To use the MTA server directly, Bob would need to run the MTA server all the time
  - Second, note that **Bob needs** another pair of client-server programs:
    - **Message access programs.**
      - This is because an MTA client-server program is a push program:
        - The client pushes the message to the server.
      - Bob needs a pull program.
        - The client needs to pull the message from the server.

Email : Protocols

- e-mail is one of those applications that needs three uses of client-server paradigms to accomplish its task.
- Figure shows these three client-server applications.
    - First and the second as Message Transfer Agents (MTAs),
    - Third as Message Access Agent (MAA).



Email : Protocols

Simple Mail Transfer Protocol (SMTP).

- Formal protocol that defines the **MTA client and server**
- SMTP is used two times,
  - Between the **sender and the sender's mail server** and
  - Between **the two mail servers**.

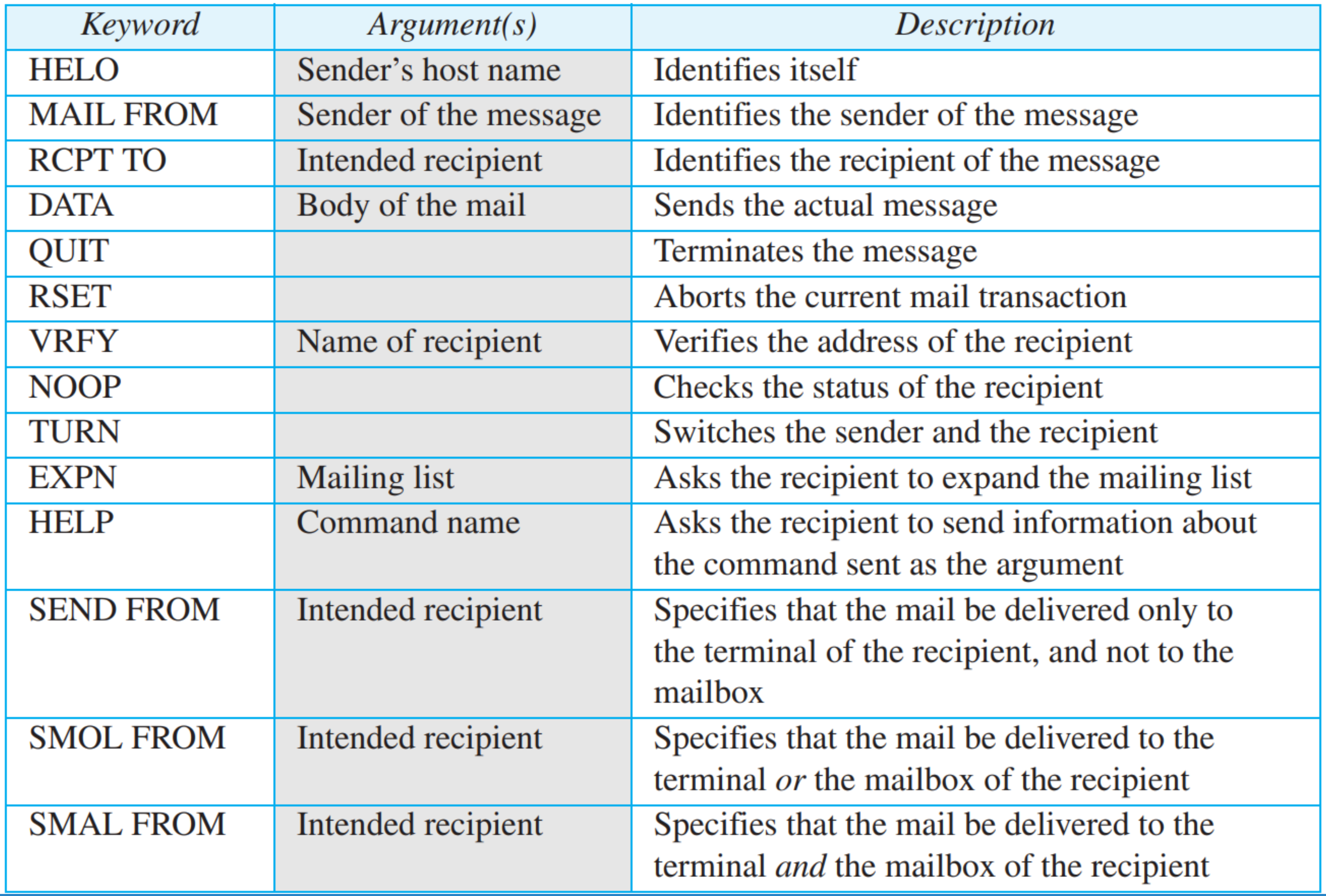
- **SMTP simply defines how commands and responses must be sent back and forth.**

- SMTP Commands and Responses
  - Used to **transfer messages between** an **MTA client** and an **MTA server**.
  - **Command** is from an MTA client to an MTA server;
  - **Response** is from an MTA server to the MTA client.
    - *Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.*

Email : Protocols

Simple Mail Transfer Protocol (SMTP).

- **SMTP Commands:** Commands are sent from the client to the server

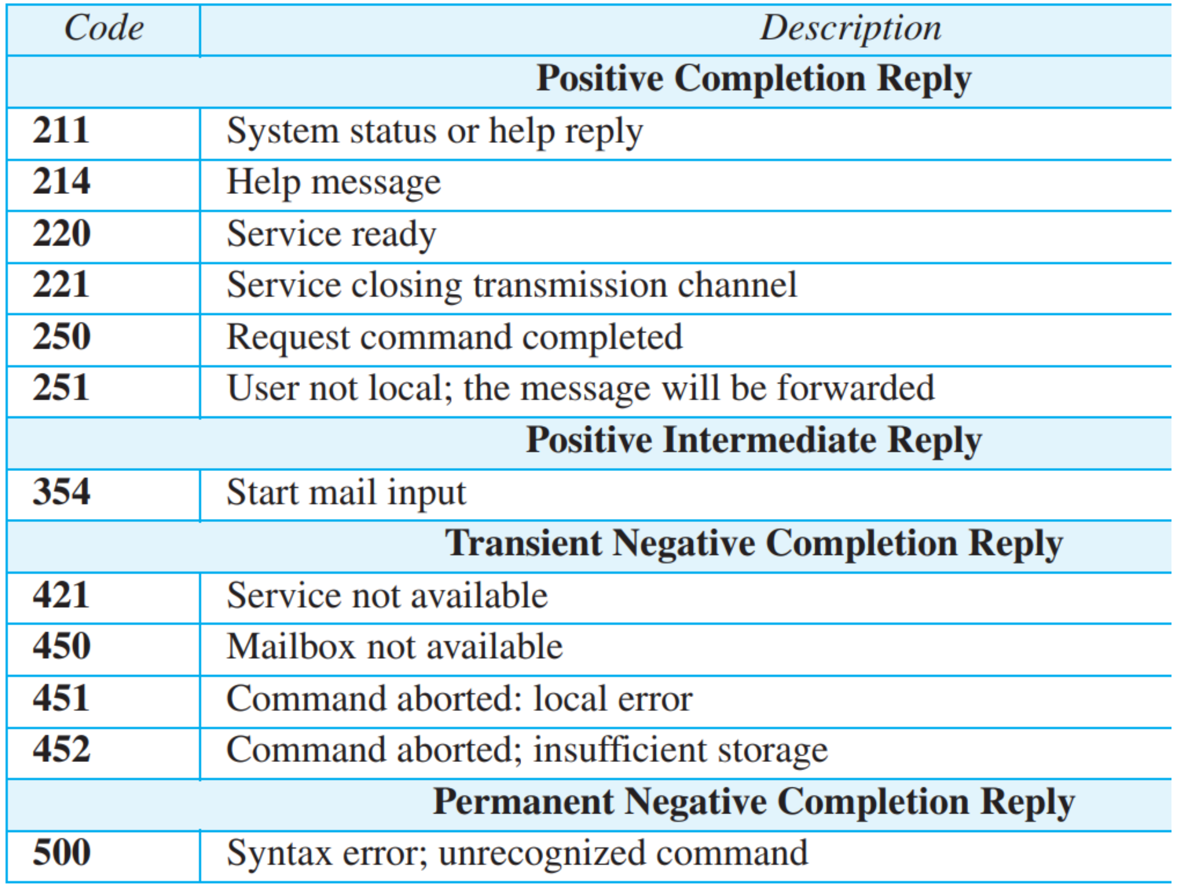


Keyword	*Argument(s)*	*Description*
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current mail transaction
VRFY	Name of recipient	Verifies the address of the recipient
NOOP		Checks the status of the recipient
TURN		Switches the sender and the recipient
EXPN	Mailing list	Asks the recipient to expand the mailing list
HELP	Command name	Asks the recipient to send information about the command sent as the argument
SEND FROM	Intended recipient	Specifies that the mail be delivered only to the terminal of the recipient, and not to the mailbox
SMOL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *or* the mailbox of the recipient
SMAL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *and* the mailbox of the recipient

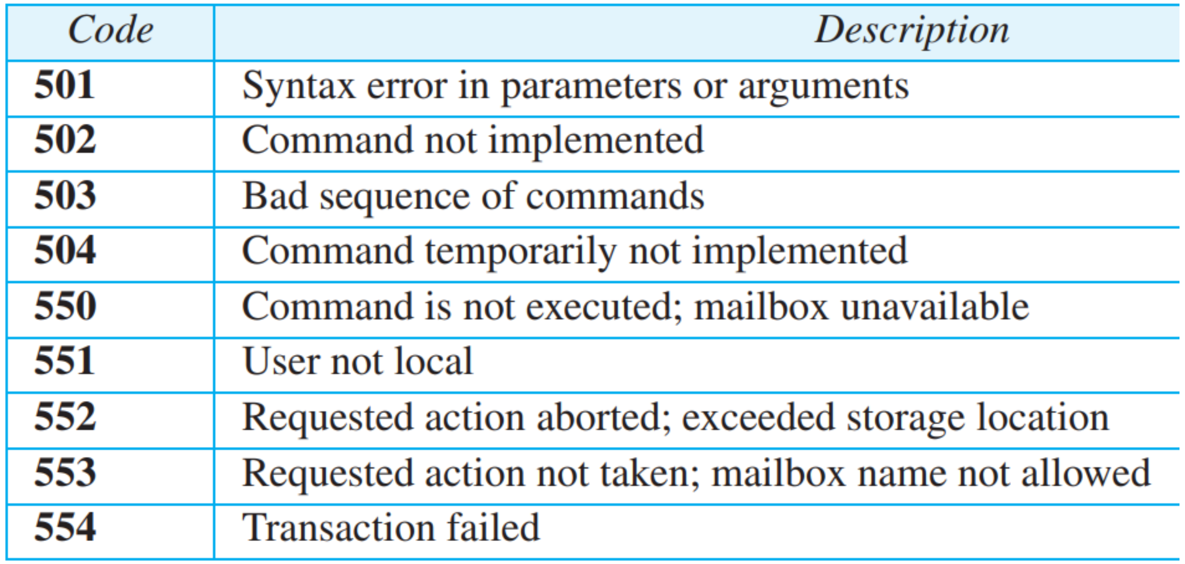
Email : Protocols

Simple Mail Transfer Protocol (SMTP).

- **SMTP Responses:** Commands are sent from the server to the client



Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command



Code	*Description*
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Email : Mail Transfer Phases

- The process of transferring a mail message occurs in three phases:
  - Connection establishment, Mail transfer, and Connection termination.

- Connection Establishment
  - After a client has made a **TCP connection to the** wellknown port 25.
  - The SMTP server starts the connection phase.
  - Follows three steps:

- Step 1 :
  - Server sends code 220 (service ready) to tell the client that it is ready to receive mail.
  - If the server is not ready, it sends code 421 (service not available).

- Step 2 :
  - The client sends the HELO message to identify itself, using its domain name address.
  - This step is necessary to inform the server of the domain name of the client.

- Step 3 :
  - erver responds with code 250 (request command completed) or some other code depending on the situation.

Email : Mail Transfer Phases

- Mail transfer
  - Involves eight steps. Steps 3 and 4 are repeated if **there is more than one recipient**.

  - Step 1. The client sends the **MAIL FROM** message to introduce the sender of the message.
  - Step 2. The server **responds with code 250** or some other appropriate code.
  - Step 3. The client sends the **RCPT TO (recipient)** message, which includes the mail address of the recipient.
  - Step 4. The server **responds with code 250** or some other appropriate code.
  - Step 5. The client sends the DATA message to initialize the message transfer.
  - Step 6. The server responds with code 354 (start mail input) or other appropriate message.
  - Step 7. The client sends the ***contents of the message in consecutive lines***. ***Each line is terminated by a two-character end-of-line token*** (carriage return and line feed). The message is terminated by a line containing just one period.
  - Step 8. The server **responds with code 250 (OK)** or some other appropriate code.

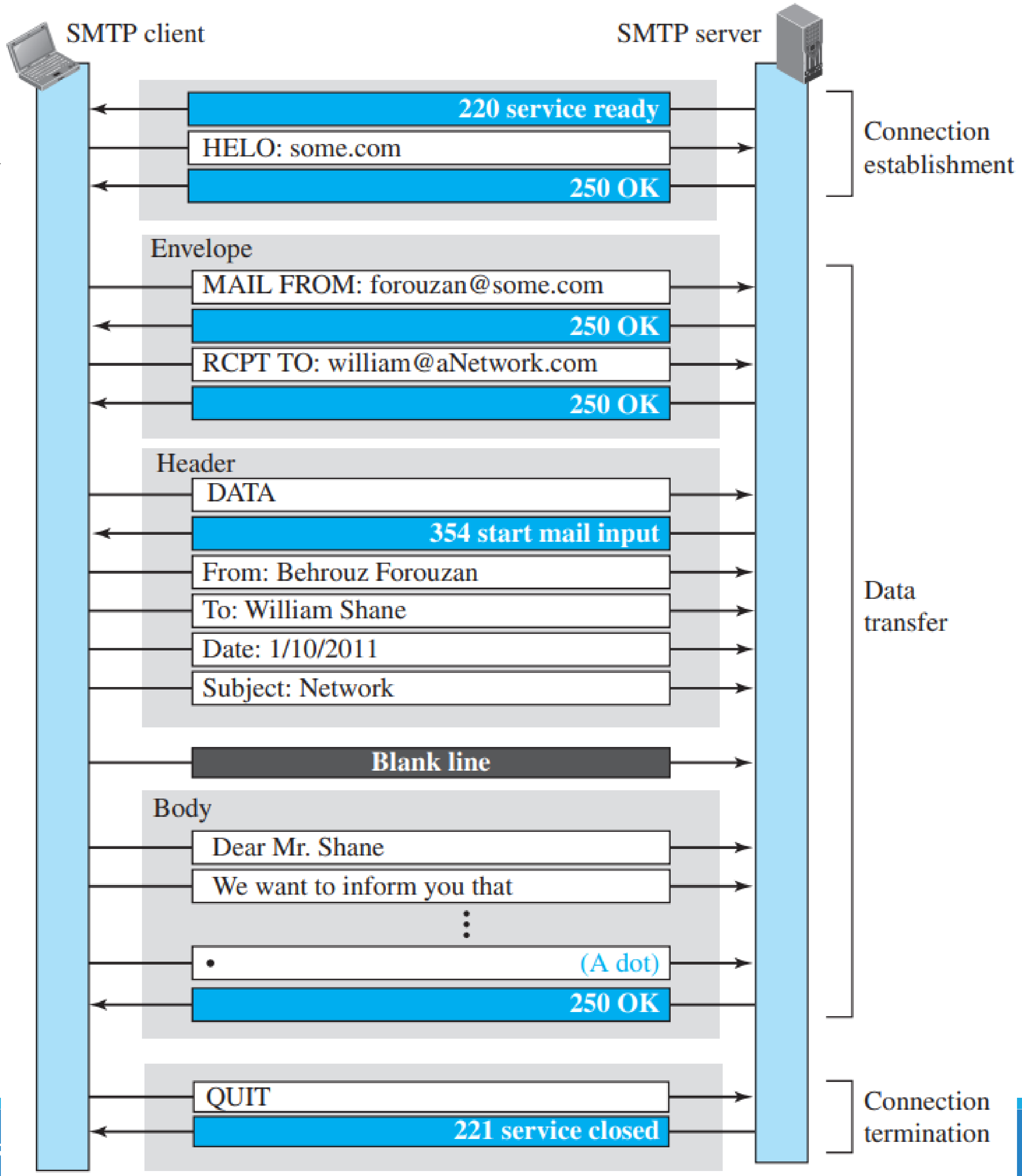
Email : Mail Transfer Phases

Connection Termination

- After the message is transferred successfully, the client terminates the connection.
- This phase involves two steps.
  - 1. The **client sends the QUIT** command.
  - 2. The server **responds with code 221** or some other appropriate code

Email :
Mail Transfer Phases

Example



Email :
Message Access Agent: POP and IMAP

- The first and second stages of mail delivery use SMTP.
- **SMTP is not involved in the third stage** because:
  - SMTP is a push protocol;
  - It pushes the message from the client to the server.
  - In other words, the direction of the bulk data is from the client to the server.

- **On the other hand, the third stage needs a pull protocol**;
  - The client must pull messages from the server.
  - The direction of the ***bulk data is from the server to the client***.
  - The third stage uses a message access agent.

- Currently two message access protocols are available:
  - Post Office Protocol, version 3 (POP3) and
  - Internet Mail Access Protocol, version 4 (IMAP4).

Email :
Message Access Agent: POP and IMAP

- The first and second stages of mail delivery use SMTP.
- **SMTP is not involved in the third stage** because:
    - SMTP is a push protocol;
    - It pushes the message from the client to the server.
    - In other words, the direction of the bulk data is from the client to the server.

- **On the other hand, the third stage needs a pull protocol**;
    - The client must pull messages from the server.
    - The direction of the ***bulk data is from the server to the client***.
    - The third stage uses a message access agent.

- Currently two message access protocols are available:
    - Post Office Protocol, version 3 (POP3) and
    - Internet Mail Access Protocol, version 4 (IMAP4).

Email :
Message Access Agent: POP

POP3

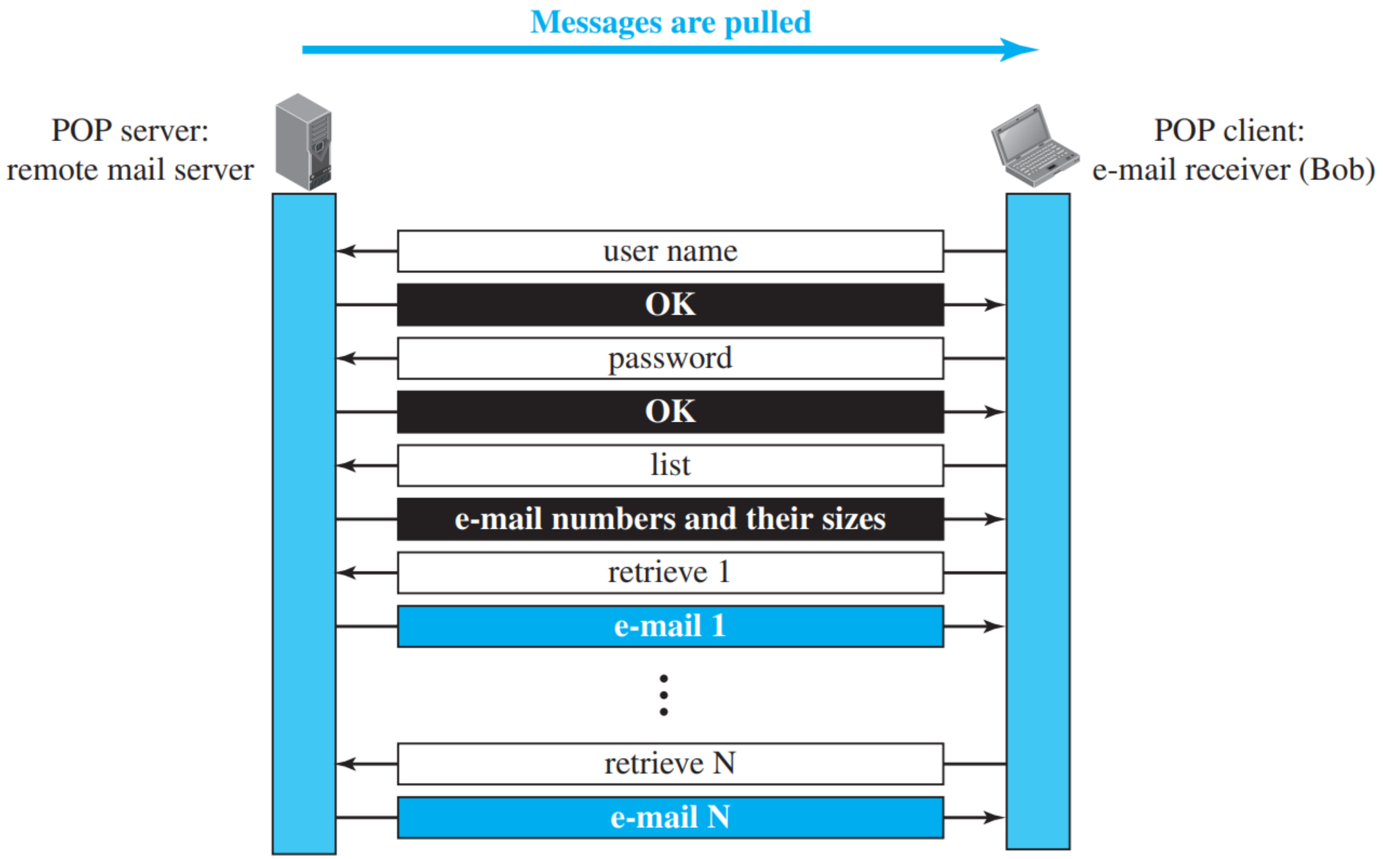
- Post Office Protocol, version 3 (POP3) is **simple** but limited in functionality.
  - The *client POP3 software* is installed on the *recipient computer*;
  - The *server POP3 software* is installed on the *mail server*.

- Mail access starts with the ***client when the user needs to download its 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.

Email :
Message Access Agent: POP

POP3

- Figure shows an example of downloading using POP3.
- The client on the right hand side because the e-mail receiver (Bob) is running the client process to pull messages from the remote mail server


Messages are pulled
POP server: remote mail server
POP client: e-mail receiver (Bob)
user name
OK
password
OK
list
e-mail numbers and their sizes
retrieve 1
e-mail 1
retrieve N
e-mail N

Email :
Message Access Agent: POP

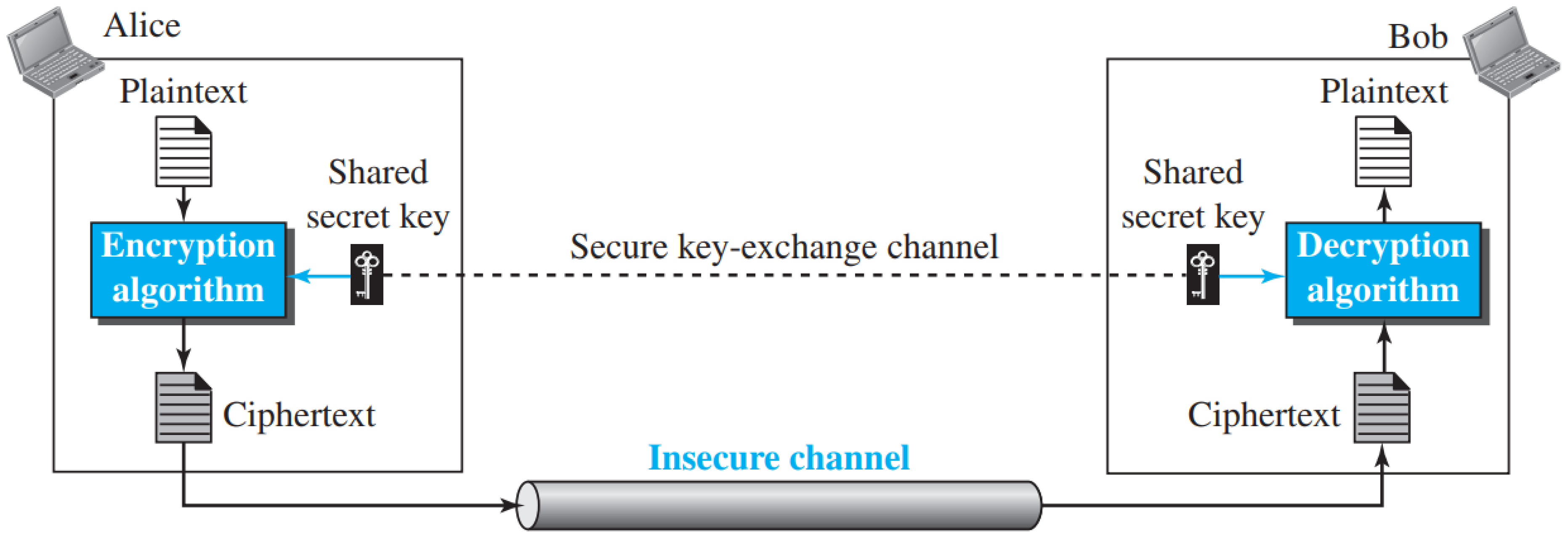
POP3 : *has two modes:*

- **Delete mode:**
  - In the delete mode, the mail is deleted from the mailbox after each retrieval.
- **Keep mode.**
  - In the keep mode, the mail remains in the mailbox after retrieval.

Cryptography

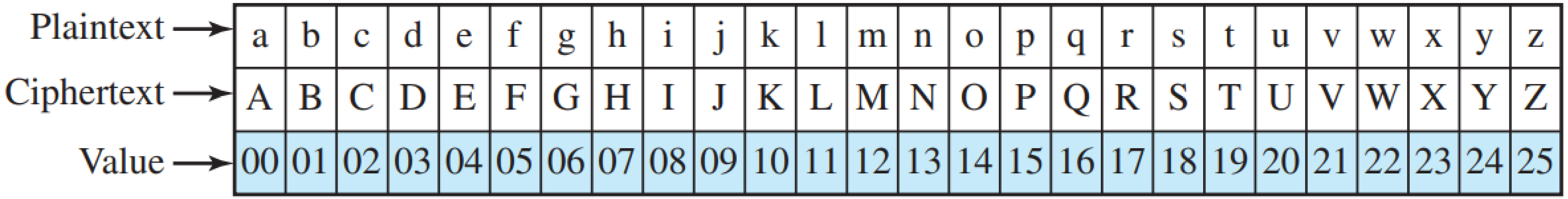
Symmetric-Key Ciphers

- A symmetric-key cipher uses the **same key for both encryption and decryption,**
- The key can be used for bidirectional communication, which is why it is called symmetric.



Cryptography

Representation of plaintext and ciphertext characters in modulo 26



Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

In additive cipher, the plaintext, ciphertext, and key are integers in modulo 26.

Cryptography

Use the additive cipher with key = 15 to encrypt the message "hello".

Solution

We apply the encryption algorithm to the plaintext, character by character:

Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

The result is "WTAAD". Note that the cipher is monoalphabetic because two instances of the same plaintext character (*l*) are encrypted as the same character (*A*).

Cryptography

Use the additive cipher with key = 15 to decrypt the message “WTAAD”.

Solution

We apply the decryption algorithm to the plaintext character by character:

Ciphertext: W $\rightarrow$ 22	Decryption: (22 – 15) mod 26	Plaintext: 07 $\rightarrow$ h
Ciphertext: T $\rightarrow$ 19	Decryption: (19 – 15) mod 26	Plaintext: 04 $\rightarrow$ e
Ciphertext: A $\rightarrow$ 00	Decryption: (00 – 15) mod 26	Plaintext: 11 $\rightarrow$ l
Ciphertext: A $\rightarrow$ 00	Decryption: (00 – 15) mod 26	Plaintext: 11 $\rightarrow$ l
Ciphertext: D $\rightarrow$ 03	Decryption: (03 – 15) mod 26	Plaintext: 14 $\rightarrow$ o

The result is “hello”. Note that the operation is in modulo 26, which means that we need to add 26 to a negative result (for example −15 becomes 11).

Cryptography

Asymmetric-Key Ciphers

- The conceptual differences between the two systems are based on how these systems keep a secret.

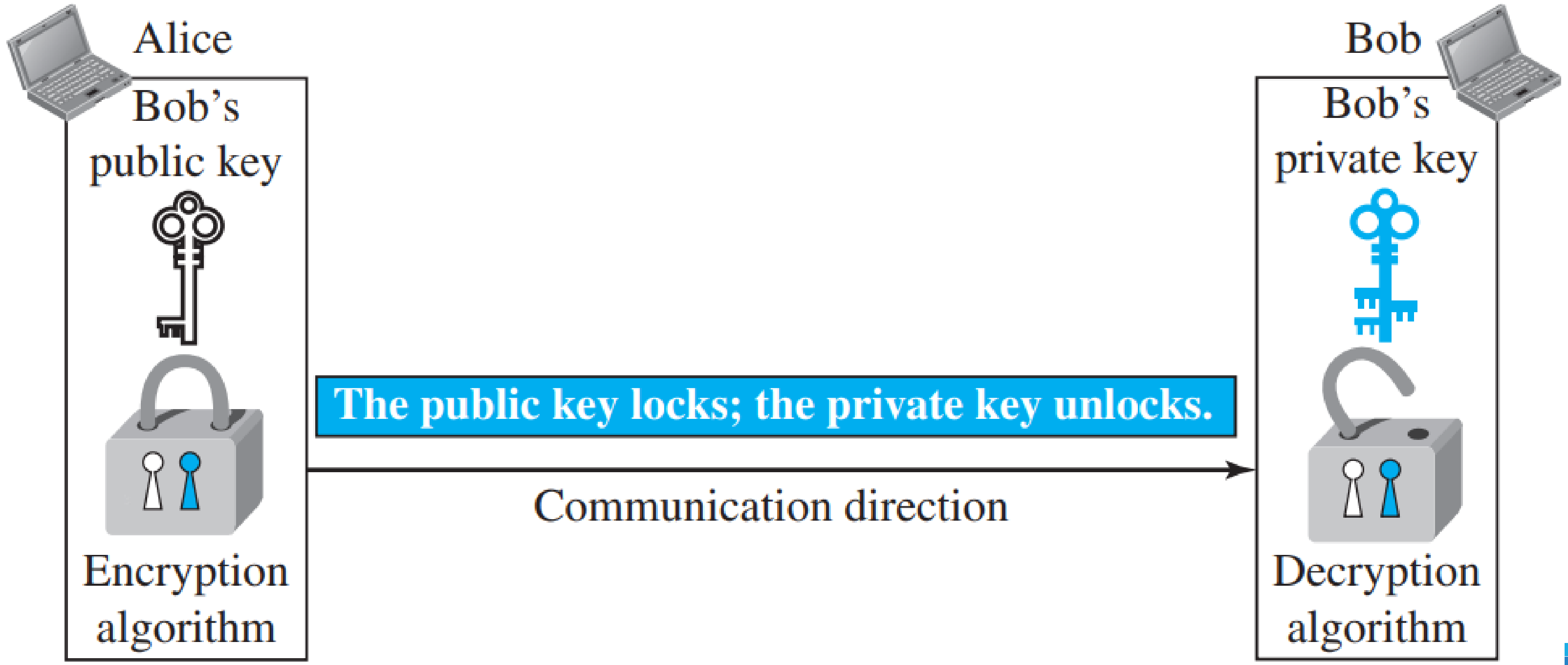
- **Symmetric-key cryptography**, the key must be shared between two persons.
- **Asymmetric-key cryptography**, the key is personal (unshared); each person creates and keeps his or her own secret
    - These include authentication and digital signatures.
    - Whenever an application is based on a personal key, we need to use asymmetric-key cryptography.

- **Symmetric-key cryptography** is based on **substitution and permutation of symbols.**
- **Asymmetric-key cryptography** is based on **applying mathematical functions to numbers**.

Cryptography

- **Asymmetric key cryptography uses two separate keys:**
  - One private and One public.

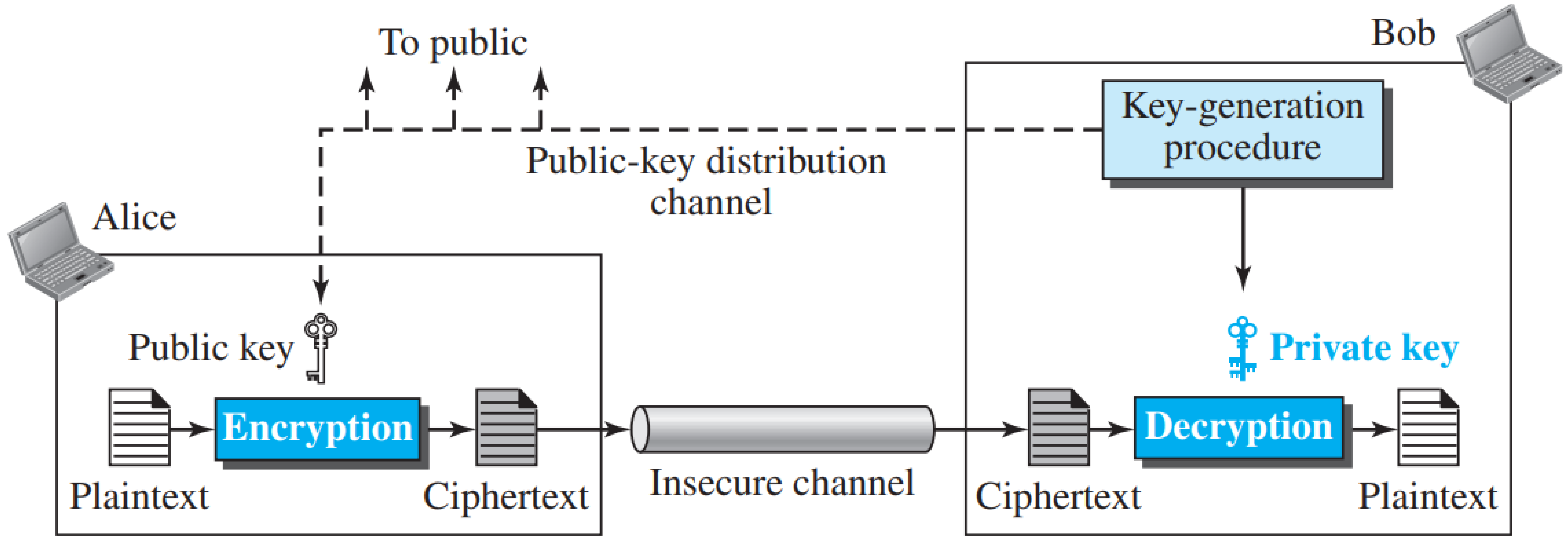
If encryption and decryption are thought of as locking and unlocking padlocks with keys, then the padlock that is locked with a public key can be unlocked only with the corresponding private key.



If Alice locks the padlock with Bob's public key, then only Bob's private key can unlock it.

Asymmetric-key Cryptography

- Unlike symmetric-key cryptography, there are distinctive keys in asymmetric-key cryptography:
  - A private key and A public key.
- Private/secret key is not interchangeable with a private key; there are **two different types of secrets**



Asymmetric-key Cryptography

Important Points:

- **First**, it emphasizes the **asymmetric nature of the cryptosystem**.
  - Providing security is mostly is the responsibility of the receiver (Bob, in this case).
  - Bob needs to create **two keys**: one private and one public.
  - Bob is responsible for **distributing the public key to the community**.
  - This can be done through a public-key distribution channel.
  - Eve should not be able to advertise her public key to the community pretending that it is Bob’s public key.

Asymmetric-key Cryptography

- **Second**, asymmetric-key cryptography means that Bob and Alice cannot use the same set of keys for two-way communication.
  - Each entity in the community should create its own private and public keys.
  - Figure shows how Alice can use Bob’s public key to send encrypted messages to Bob.
  - If Bob wants to respond, Alice needs to establish her own private and public keys.

- **Third**, asymmetric-key cryptography means that Bob needs only one private key to receive all correspondence from anyone in the community,
  - But Alice needs n public keys to communicate with n entities in the community, one public key for each entity. In other words, Alice needs a ring of public keys.

Cryptography

Plaintext/Ciphertext

- Unlike in symmetric-key cryptography, **plaintext and ciphertext in asymmetric-key cryptography are treated as integers.**
  - **Message must be encoded as an integer** (or a set of integers) **before encryption**;
  - **The integer** (or the set of integers) must be **decoded into the message** after **decryption**.
- Asymmetric-key cryptography is normally used to ***encrypt or decrypt small pieces of information***, such as the ***cipher key for a symmetric-key cryptography***.

Encryption/Decryption

- Are **mathematical functions applied over the numbers representing the plaintext and ciphertext.**
- The ciphertext can be thought of as $\boldsymbol{C = f\,(K_{public}, P)}$;
- The plaintext can be thought of as $P = g\,(K_{private}, C)$.
- The encryption function $\boldsymbol{f}$ is used only for encryption;
- The decryption function $\boldsymbol{g}$ is used only for decryption.

RSA Cryptosystem

Although there are several asymmetric-key cryptosystems, one of the common public-key algorithms is the RSA cryptosystem, named for its inventors Rivest, Shamir, and Adleman.

RSA uses two exponents,

- ***e*** and ***d***, where ***e is public*** and ***d is private***.

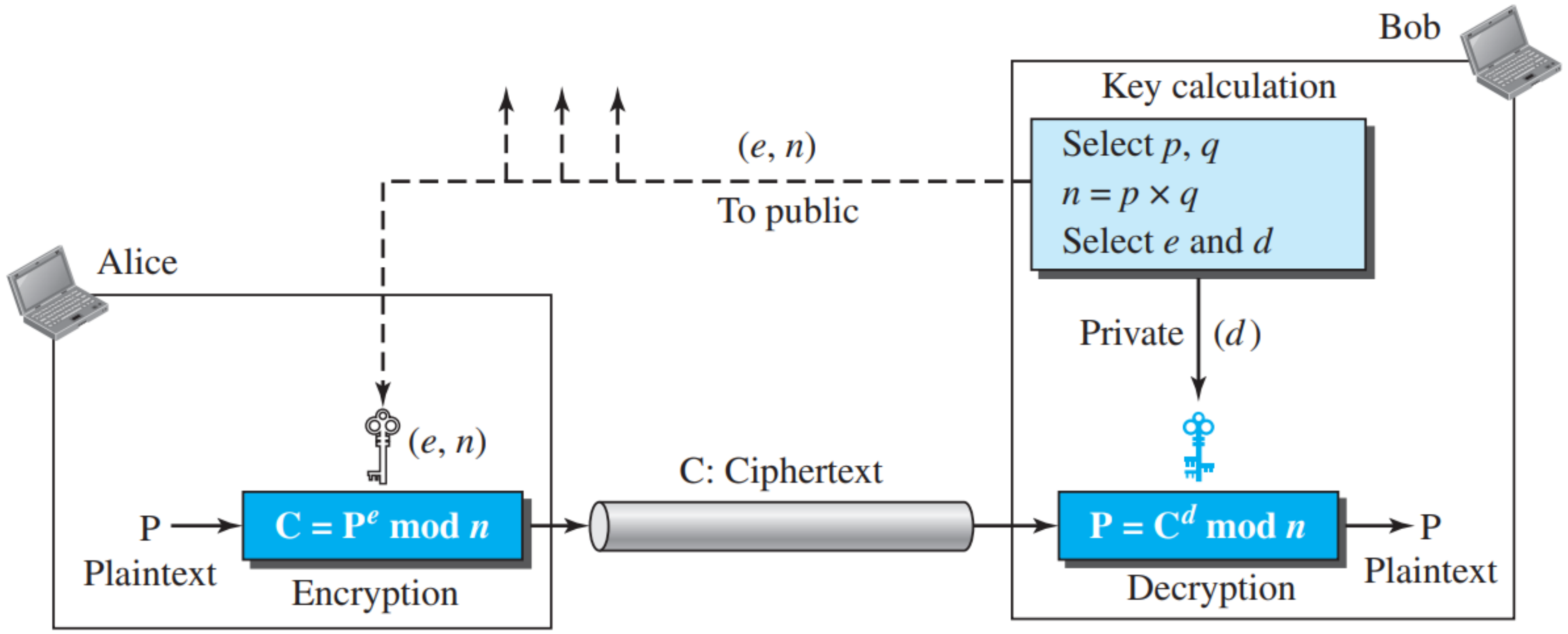
Suppose P is the plaintext and C is the ciphertext.

- Alice uses $\boldsymbol{C = P^e \bmod n}$ to create **cipher-text C** from plaintext P;
- Bob uses $\boldsymbol{P = C^d \bmod n}$ to **retrieve the plaintext** sent by Alice.
- The **modulus n**, a very large number, is created during the key generation process.

RSA Cryptosystem

RSA Procedure:

- Bob chooses **two large numbers, p and q**, and calculates
  - $n = p \times q$
  - $\varphi = (p - 1) \times (q - 1)$.
- Bob then selects e and d such that $(e \times d) \bmod \varphi = 1$.



RSA Cryptosystem

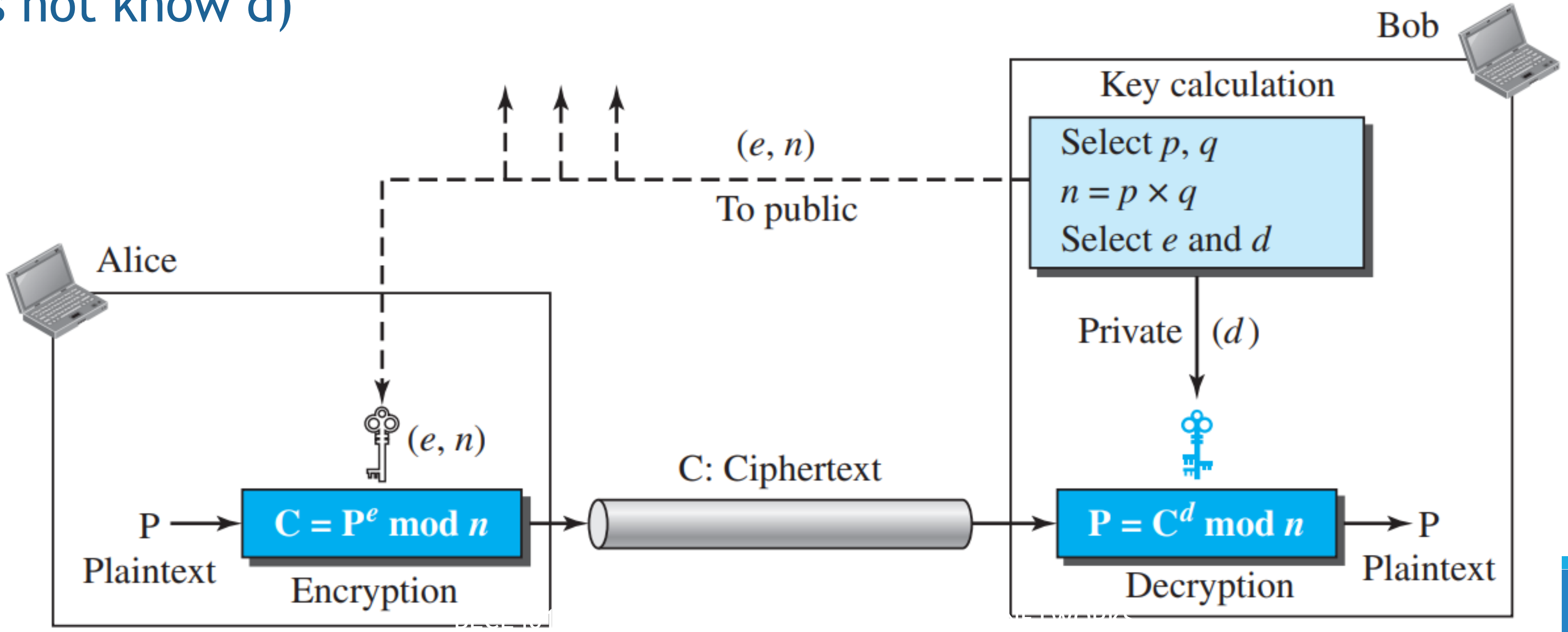
Bob advertises e and n to the community as the public key;

- Bob keeps d as the private key.

Anyone, including Alice, can encrypt a message and send the ciphertext to Bob, using $C = (Pe)\ mod\ n$;

only Bob can decrypt the message, using $P = (C^d)\ mod\ n$.

An intruder such as Eve cannot decrypt the message if p and q are very large numbers (she does not know d)



RSA Cryptosystem

Step 1: Key Generation

- Choose two prime numbers, p & q
- Compute $n = p \times q$ {part of public key}
- Compute $\phi(n) = (p-1) \times (q-1)$
- Choose an integer e such that $1 < e < \phi(n)$
  - Also e is co-prime to $\phi(n)$ i.e. $GCD\left(e, \phi(n)\right) = 1$
  - e forms the other part of public key
- Find d such that $e \times d \bmod \phi(n) = 1$

Step 2: Encryption

- Encrypt a **Message P** {usually a number}, sender uses the **recipient's public keys {e,n}**
- Determine **Encrypted Message** as $\boldsymbol{C = (P^e) \bmod n}$

Step 3: Decryption

- **Encrypted Message C** is decrypted by the recipient by using **private key d**
- **Decrypted message** can be found using $\boldsymbol{P = (C^d) \bmod n}$

RSA Cryptosystem

Example 1:

Step 1: Key Generation

- Two prime numbers, 3 & 5
- Compute $n = 3 \times 5 = \mathbf{15}$ {part of public key}
- Compute $\phi(n) = (3-1) \times (5-1) = \mathbf{8}$
- Chose e, such that $\boldsymbol{GCD\ (e,\phi(n))=1}$ & $\mathbf{1<e<\phi(n)}$
  - $\gcd(3,8) = 1, \quad \gcd(5,8) = 1, \quad \gcd(6,8) = 2$
  - e = 3
- Find d such that $e \times d \bmod \phi(n) = 1$
  - $3 \times d \bmod 8 = 1 \Rightarrow \boldsymbol{d = 3}$

Public Key {3,15}

Private Key {3, 15}

Step 2: Encryption

- Consider **Message P = 8**
- So **Encrypted Message:**
  - $\boldsymbol{C = (P^e) mod\ n} = \mathbf{(8^3) mod\ 15}$
  - $\mathbf{C = 512\ mod\ 15 = 2}$

Step 3: Decryption

- **Decrypted message**
  - $\boldsymbol{P = (2^3) mod\ 15}$
  - **P = 8**