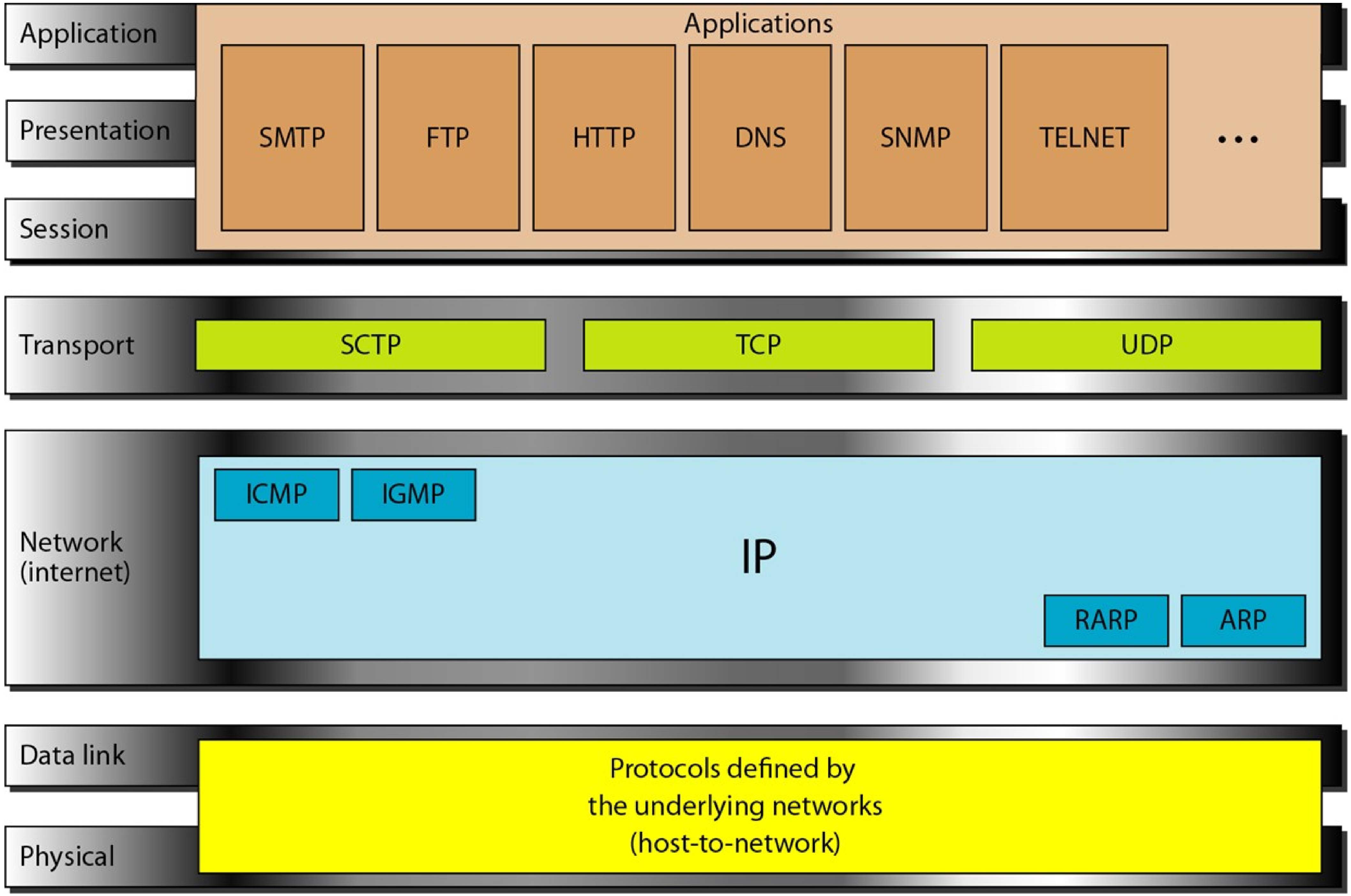
Computer Networks

Application Layer – TCP/IP Model

Lecture # 03

TCP/IP Protocol Suite

- The layers in the TCP/IP protocol suite do not exactly match those in the OSI model.
- The original TCP/IP protocol suite was defined as having four layers: host-to-network, internet, transport, and application.
- However, when TCP/IP is compared to OSI, we can say that the TCP/IP protocol suite is made of five layers:
  - physical,
  - data link,
  - network,
  - transport, and
  - application.


Application
Presentation
Session
Applications
SMTP
FTP
HTTP
DNS
SNMP
TELNET
• • •
Transport
SCTP
TCP
UDP
Network
(internet)
ICMP
IGMP
IP
RARP
ARP
Data link
Physical
Protocols defined by
the underlying networks
(host-to-network)

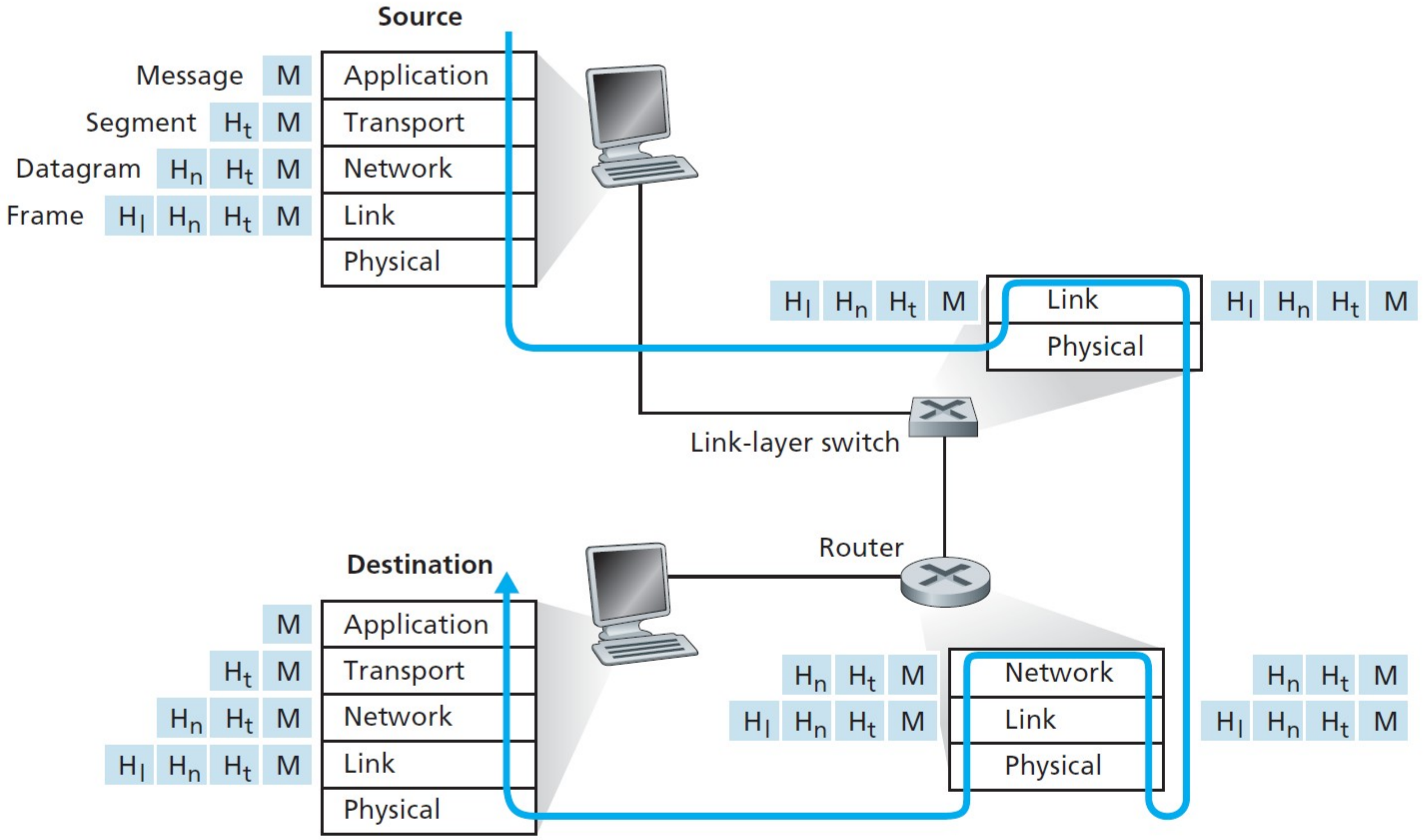
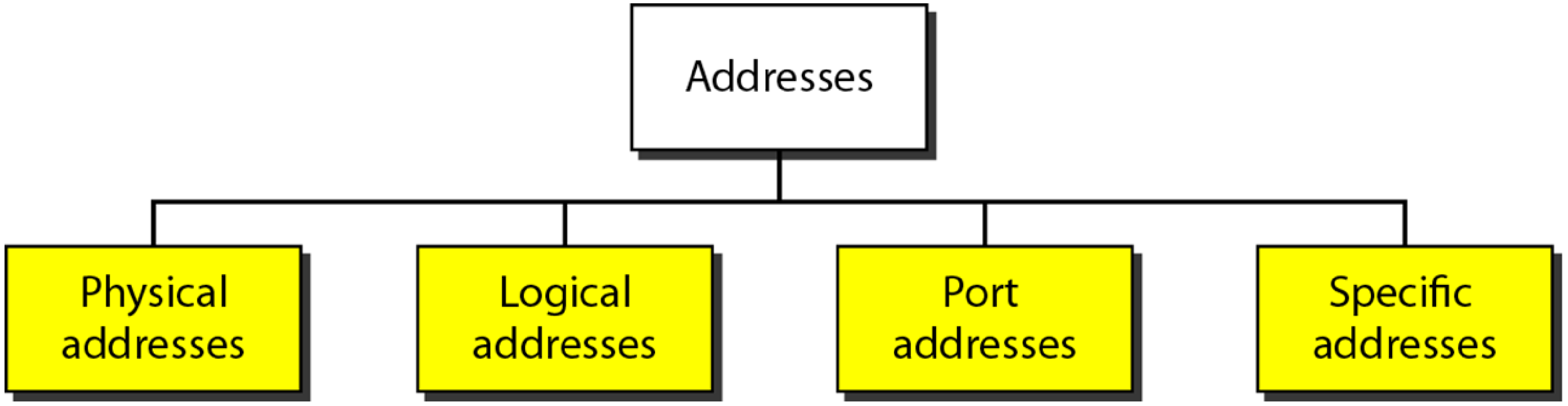
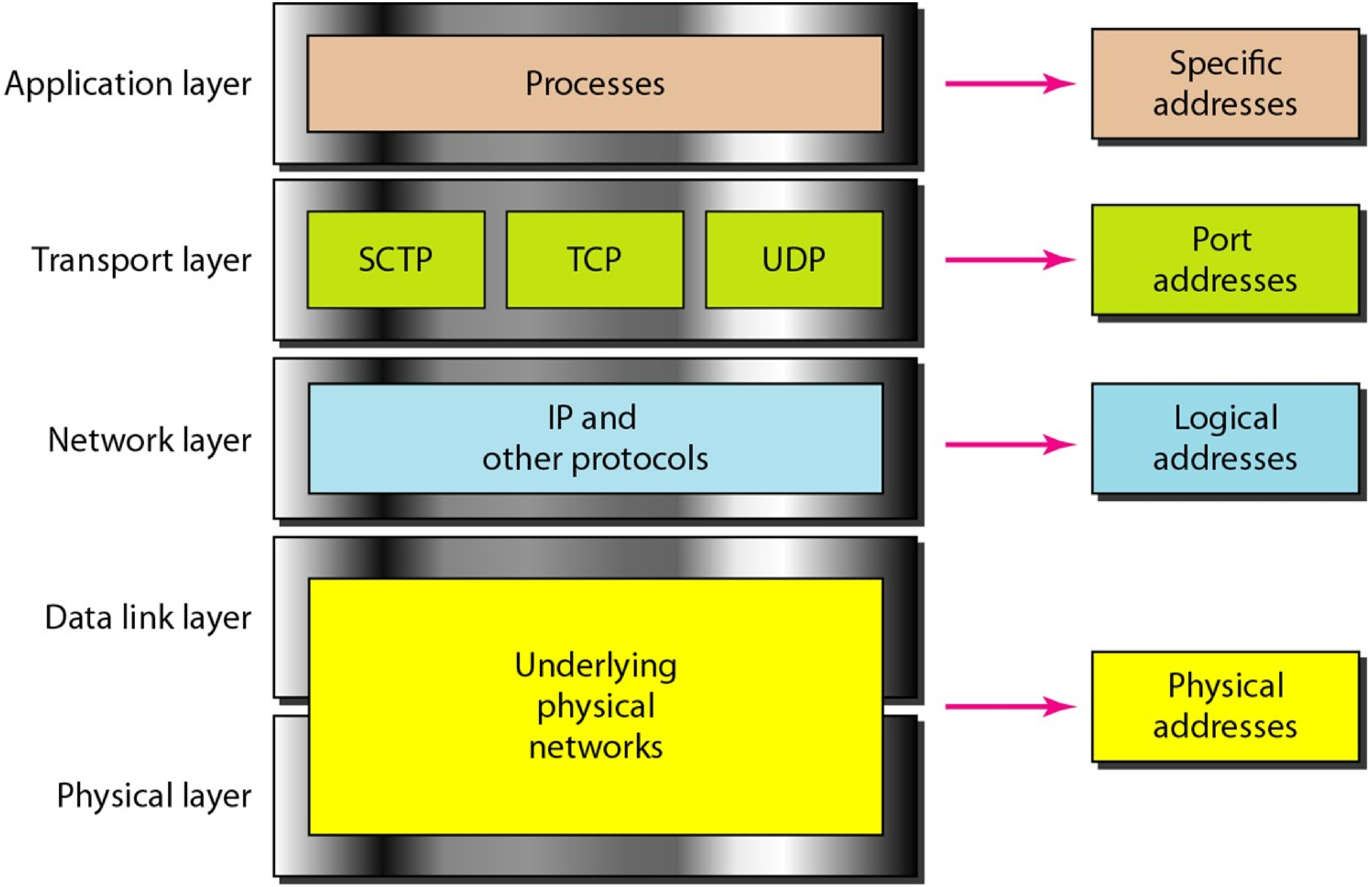


Figure 1.24 ♦ Hosts, routers, and link-layer switches; each contains a different set of layers, reflecting their differences in functionality

Addressing

- Four levels of addresses are used in an internet employing the *TCP/IP* protocols:
  - physical (link) addresses,
  - logical (IP) addresses,
  - port addresses, and
  - specific addresses




Application layer
Processes
Specific addresses
Transport layer
SCTP
TCP
UDP
Port addresses
Network layer
IP and other protocols
Logical addresses
Data link layer
Physical layer
Underlying physical networks
Physical addresses

Application Layer

- The conceptual and implementation aspects of network applications.
- Key application-layer concepts, including
  - network services required by applications,
  - clients and servers,
  - processes, and
  - transport-layer interfaces.
- Several network applications in detail, including the Web, e-mail, DNS, and peer-to-peer (P2P) file distribution, multimedia applications, including streaming video and VoIP.
- Network application development, over both TCP and UDP. In-particular, the socket API and walk through some simple client-server applications

Layer 7 – Application Layer

- The application layer enables the user, whether human or software, to access the network.
- It provides user interfaces and support for services such as electronic mail, remote file access and transfer, shared database management, and other types of distributed information services.
- The application layer is responsible for providing services to the user.

- **Network virtual terminal:** A network virtual terminal is a software version of a physical terminal, and it allows a user to log on to a remote host. To do so, the application creates a software emulation of a terminal at the remote host. The user's computer talks to the software terminal which, in turn, talks to the host, and vice versa. The remote host believes it is communicating with one of its own terminals and allows the user to log on.
- **File transfer, access, and management:** This application allows a user to access files in a remote host (to make changes or read data), to retrieve files from a remote computer for use in the local computer, and to manage or control files in a remote computer locally.
- **Mail services:** This application provides the basis for e-mail forwarding and storage.
- **Directory services:** This application provides distributed database sources and access for global information about various objects and services.

Network Applications

- Program Running on end systems.
- No concern with underlying networks
- Software written in C, Java, Python etc running on multiple end devices.
- For example:
  - in the Web application there are two distinct programs that communicate with each other: the browser program running in the user’s host (desktop, laptop, tablet, smartphone, and so on); and the Web server program running in the Web server host.
  - As another example, in a P2P file-sharing system there is a program in each host that participates in the file-sharing community. In this case, the programs in the various hosts may be similar or identical.

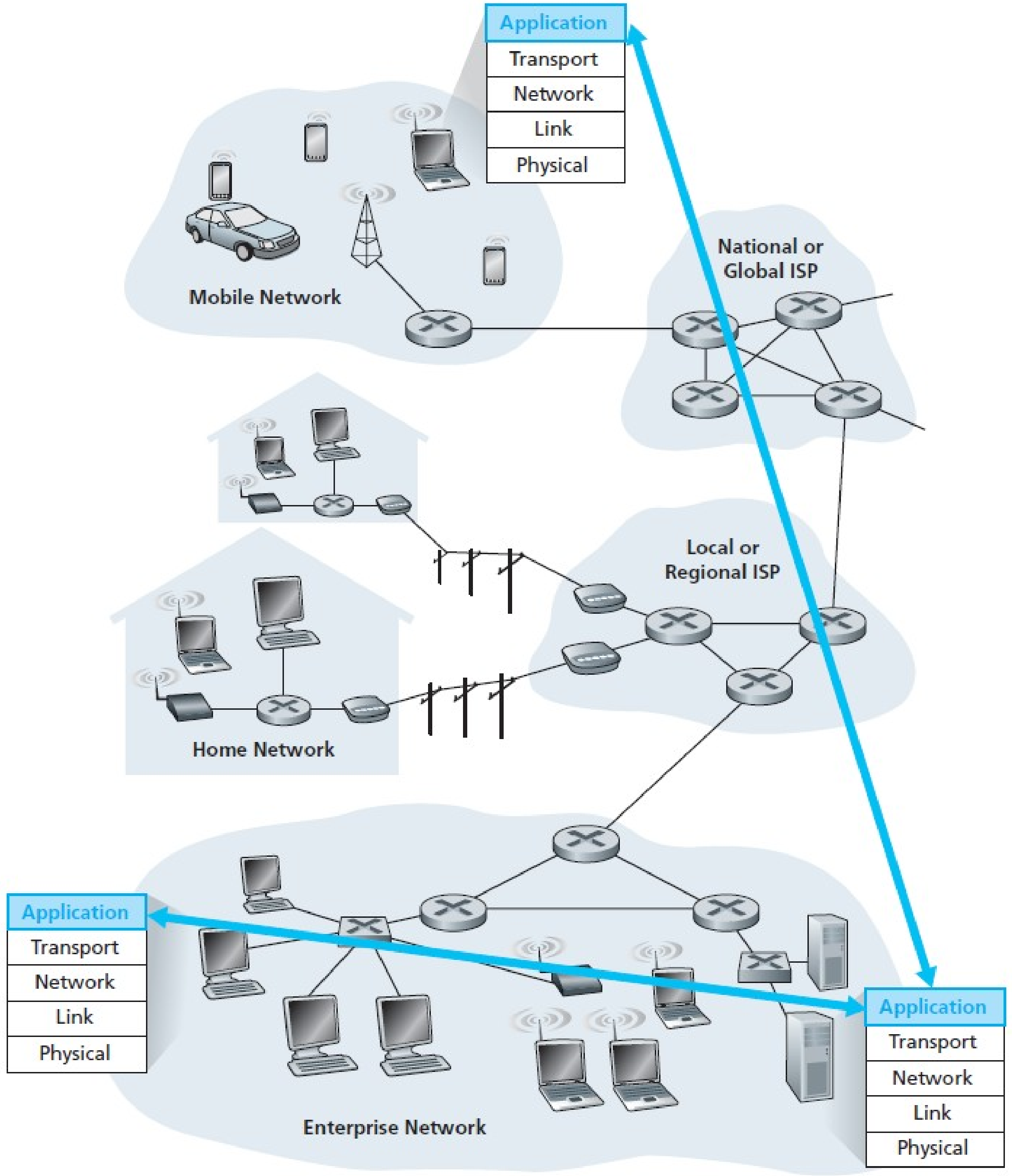


Figure 2.1 ♦ Communication for a network application takes place between end systems at the application layer

Network Application Architecture

- Two predominant architectural paradigms used in modern network applications:
  - The client-server architecture or
  - The peer-to-peer (P2P) architecture

Client-Server Architecture

- In a **client-server architecture**,
  - There is an *always-on* host, called the server, which services requests from many other hosts, called clients.
  - Another characteristic is that the server has a fixed, well-known address, called an IP address.
- Popular social-networking site can quickly become overwhelmed if it has only one server handling all of its requests. For this reason, a **data center**, housing a large number of hosts, is often used to create a powerful virtual server e.g. Google, Amazon, Facebook etc
- Examples: Web, FTP, Telnet, and e-mail are client-server.

Peer-to-Peer P2P Architecture

- Minimal (or no) reliance on dedicated servers in data centers.
- Application exploits direct communication between pairs of intermittently connected hosts, called peers.
- The peers are not owned by the service provider.
- Self scalabale.
- Many of today’s most popular and traffic-intensive applications are based on P2P architectures.
- These applications include file sharing (e.g., BitTorrent), peer-assisted download acceleration (e.g., Xunlei), Internet Telephony (e.g., Skype), and IPTV (e.g., Kankan and PPstream)

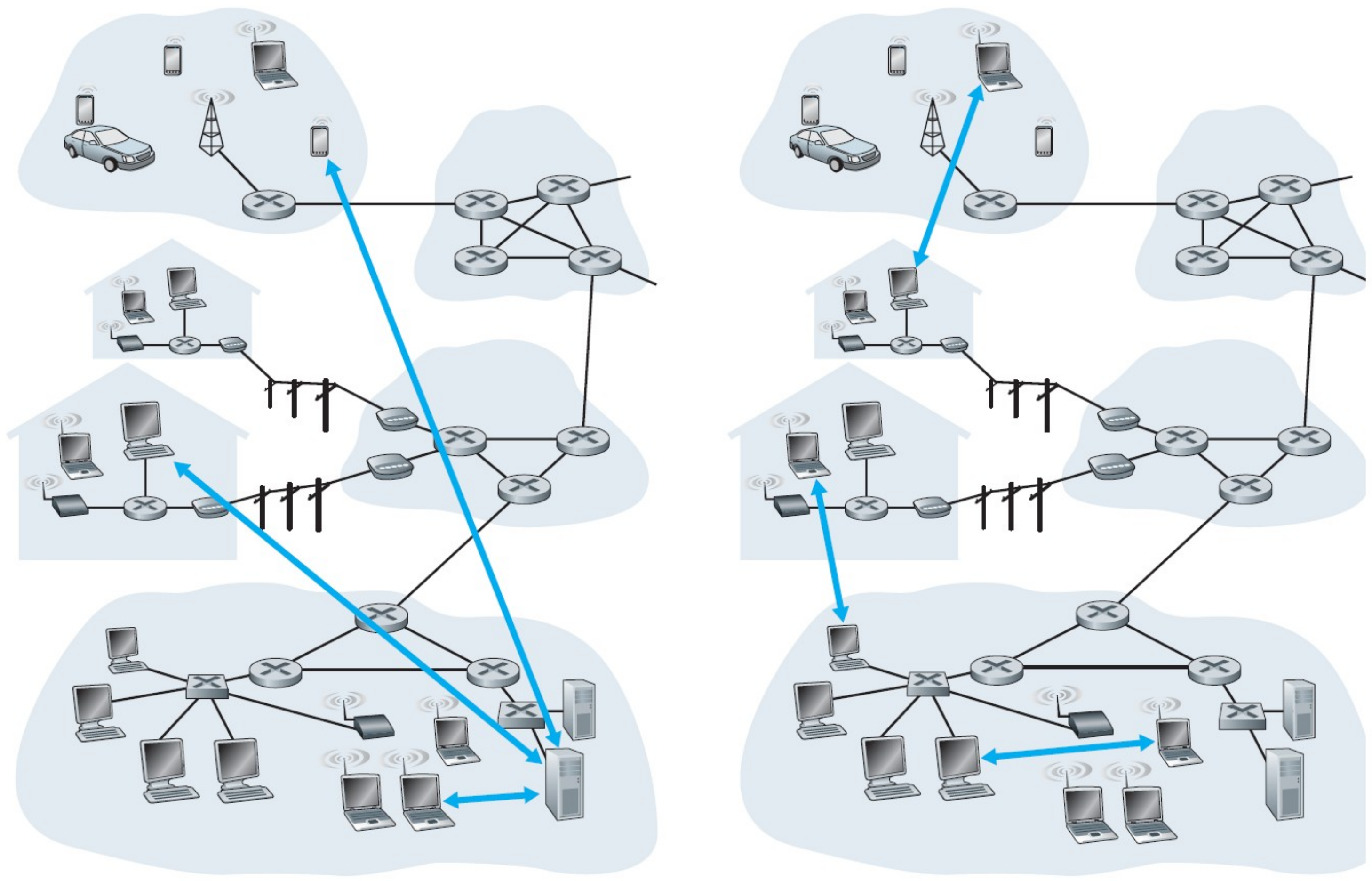


Figure 2.2 ♦ (a) Client-server architecture; (b) P2P architecture

Hybrid Architecture

- Some applications have hybrid architectures, combining both client-server and P2P elements.
- For example, for many instant messaging applications, servers are used to track the IP addresses of users, but user-to-user messages are sent directly between user hosts (without passing through intermediate servers).

P2P Chalanges

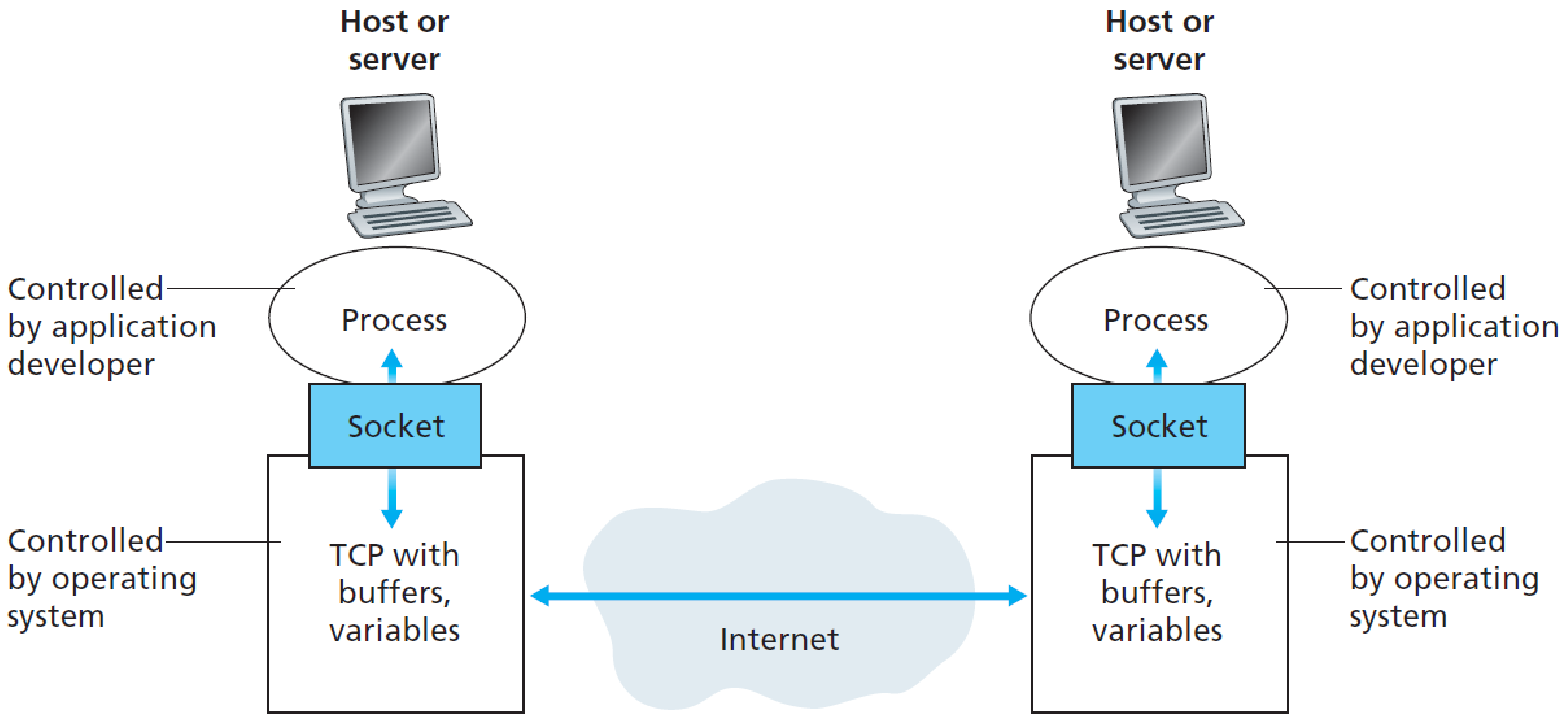
However, future P2P applications face three major challenges:

- **ISP Friendly:** Most residential ISPs (including DSL and cable ISPs) have been dimensioned for “asymmetrical” bandwidth usage, that is, for much more downstream than upstream traffic. But P2P video streaming and file distribution applications shift upstream traffic from servers to residential ISPs, thereby putting significant stress on the ISPs. Future P2P applications need to be designed so that they are friendly to ISPs
- **Security:** Because of their highly distributed and open nature, P2P applications can be a challenge to secure
- **Incentives:** The success of future P2P applications also depends on convincing users to volunteer bandwidth, storage, and computation resources to the applications, which is the challenge of incentive design

- In the context of a communication session between a pair of processes, the process that initiates the communication (that is, initially contacts the other process at the beginning of the session) is labeled as the **client**. The process that waits to be contacted to begin the session is the **server**.

Sockets

- Most applications consist of pairs of communicating processes, with the two processes in each pair sending messages to each other. Any message sent from one process to another must go through the underlying network.
- A process sends messages into, and receives messages from, the network through a software interface called a **socket.**
- A socket is the interface between the application layer and the transport layer within a host. It is also referred to as **the Application Programming Interface (API)** between the application and the network, since the socket is the programming interface with which network applications are built.
- The application developer has control of everything on the application-layer side of the socket but has little control of the transport-layer side of the socket.
- The only control that the application developer has on the transport-layer side is (1) the choice of transport protocol and (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes.



Host or server
Host or server
Controlled by application developer
Process
Process
Controlled by application developer
Socket
Socket
Controlled by operating system
TCP with buffers, variables
Internet
TCP with buffers, variables
Controlled by operating system

Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

Process Address

- Similarly, in order for a process running on one host to send packets to a process running on another host, the receiving process needs to have an address.
- To identify the receiving process, two pieces of information need to be specified:
  - The address of the host and
  - An identifier that specifies the receiving process in the destination host.

Services provided to the application

- Reliable data transfer by transport layer.
- Loss tolerant applications – multimedia applications
- Bandwidth sensitive applications – multimedia applications
- Elastic applications - Electronic mail, file transfer, and Web transfers
- Security – Confidentiality, data integrity and end point authentication.

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Figure 2.4 ♦ Requirements of selected network applications

FOCUS ON SECURITY

SECURING TCP

Neither TCP nor UDP provide any encryption—the data that the sending process passes into its socket is the same data that travels over the network to the destination process. So, for example, if the sending process sends a password in cleartext (i.e., unencrypted) into its socket, the cleartext password will travel over all the links between sender and receiver, potentially getting sniffed and discovered at any of the intervening links. Because privacy and other security issues have become critical for many applications, the Internet community has developed an enhancement for TCP, called **Secure Sockets Layer (SSL)**. TCP-enhanced-with-SSL not only does everything that traditional TCP does but also provides critical process-to-process security services, including encryption, data integrity, and end-point authentication. We emphasize that SSL is not a third Internet transport protocol, on the same level as TCP and UDP, but instead is an enhancement of TCP, with the enhancements being implemented in the application layer. In particular, if an application wants to use the services of SSL, it needs to include SSL code (existing, highly optimized libraries and classes) in both the client and server sides of the application. SSL has its own socket API that is similar to the traditional TCP socket API. When an application uses SSL, the sending process passes cleartext data to the SSL socket; SSL in the sending host then encrypts the data and passes the encrypted data to the TCP socket. The encrypted data travels over the Internet to the TCP socket in the receiving process. The receiving socket passes the encrypted data to SSL, which decrypts the data. Finally, SSL passes the cleartext data through its SSL socket to the receiving process. We'll cover SSL in some detail in Chapter 8.

Application Layer Protocols

An application-layer protocol defines how an application’s processes, running on different end systems, pass messages to each other. In particular, an application-layer protocol defines:

- The types of messages exchanged, for example, request messages and response messages
- The syntax of the various message types, such as the fields in the message and how the fields are delineated.
- The semantics of the fields, that is, the meaning of the information in the fields
- Rules for determining when and how a process sends messages and responds to messages

Protocols

- Some application-layer protocols are specified in RFCs and are therefore in the public domain. E.g. Web’s application-layer protocol, HTTP (the HyperText Transfer Protocol [RFC 2616]).
- Many other application-layer protocols are proprietary and intentionally not available in the public domain. For example, Skype uses proprietary application-layer protocols.

Network Application

- Application protocol is a part of Network Application.
- The Web is a client-server application that allows users to obtain documents from Web servers on demand.
- The Web application consists of many components, including
  - a standard for document formats (that is, HTML),
  - Web browsers (for example, Firefox and Microsoft Internet Explorer),
  - Web servers (for example, Apache and Microsoft servers), and
  - an application-layer protocol.
- The Web's application-layer protocol, HTTP, defines the format and sequence of messages exchanged between browser and Web server. Thus, HTTP is only one piece (an important piece) of the Web application.

Network Applications

Some important network applications are:

- The Web – HTTP
- file transfer - FTP
- electronic mail – SMTP, POP
- directory service - DNS
- P2P applications – Distributed file sharing and lookup service

The Web- HTTP

- The HyperText Transfer Protocol (HTTP), the Web’s application-layer protocol, is at the heart of the Web.
- It is defined in [RFC 1945] and [RFC 2616].
- HTTP is implemented in two programs: a client program and a server program.
- The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and how the client and server exchange the messages.

- A Web page (also called a document) consists of objects.
- An object is simply a file—such as an HTML file, a JPEG image, a Java applet, or a video clip—that is
- addressable by a single URL.
- Each URL has two components: the hostname of the server that houses the object and the object’s path name. For example, the URL
  - http://www.someSchool.edu/someDepartment/picture.gif
- has **www.someSchool.edu** for a hostname and **/someDepartment/ picture.gif** for a path name.

- Web Browsers – clients
- Web Servers – Servers

- HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients.
- HTTP uses TCP as its underlying transport protocol (rather than running on top of UDP).
- HTTP is a stateless protocol as it does not store information about the client.

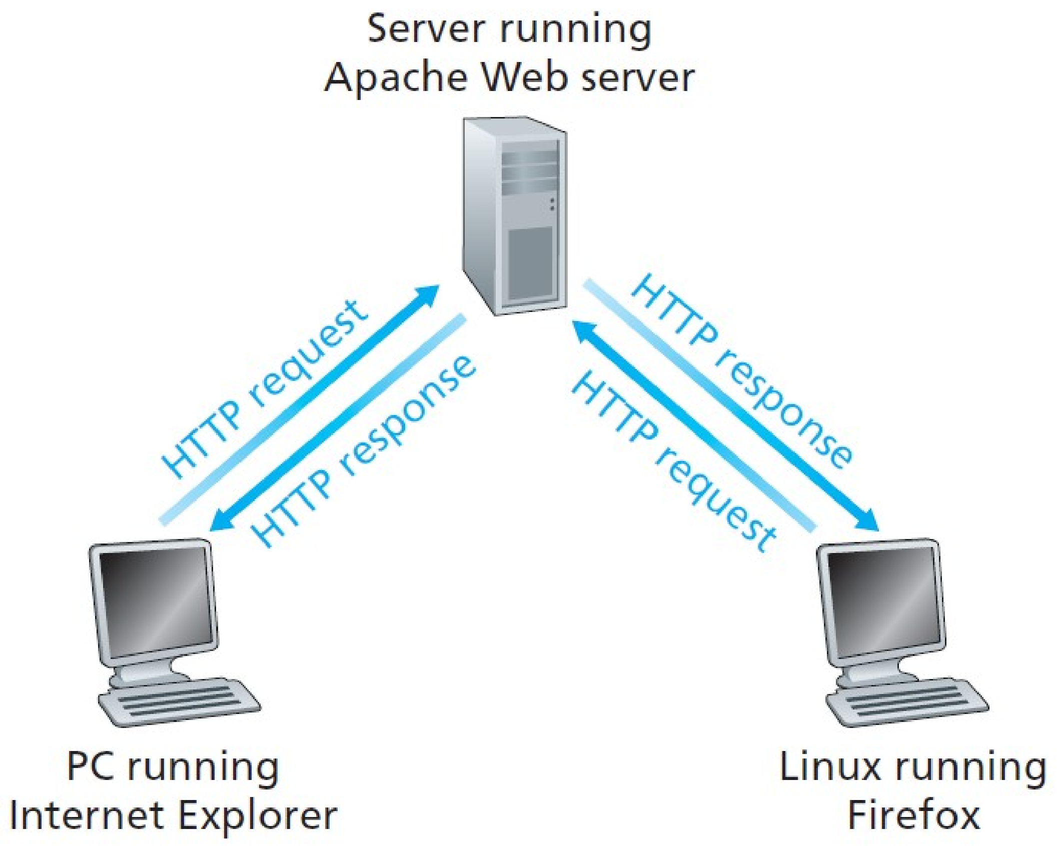


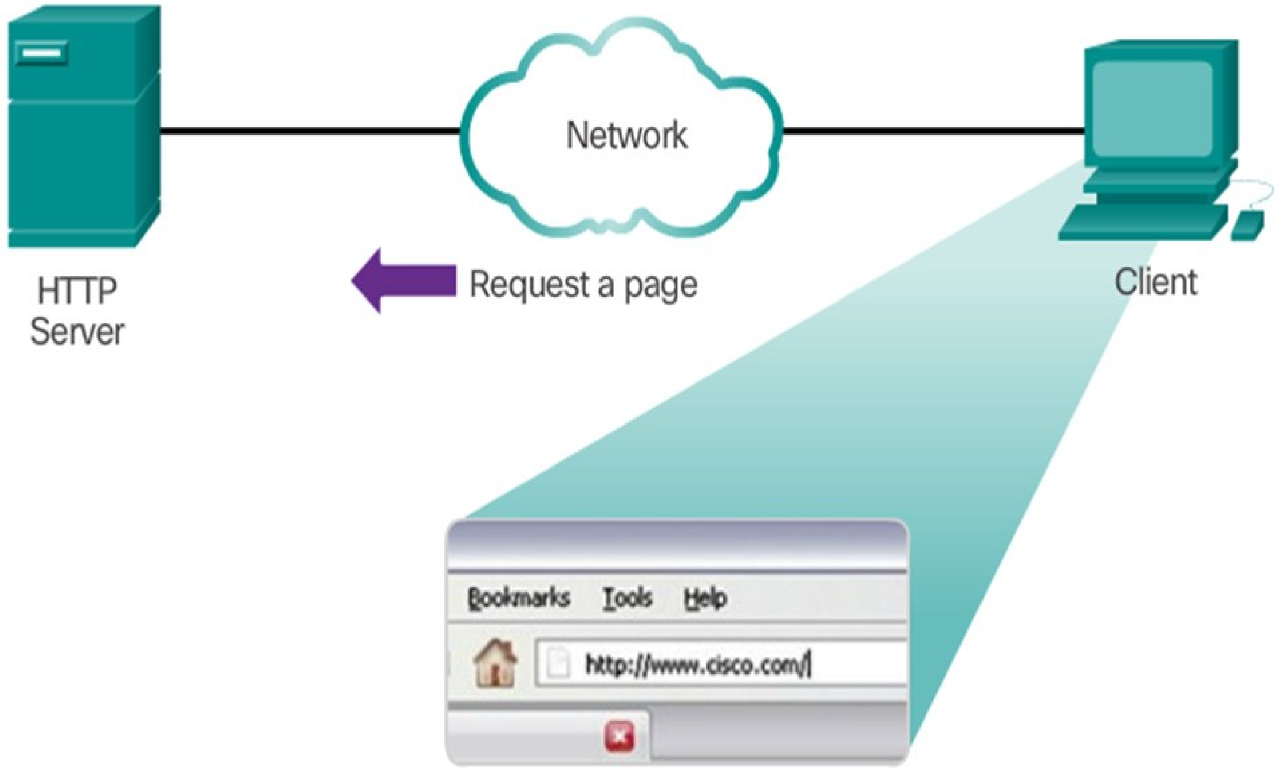
Figure 2.6 ♦ HTTP request-response behavior

Persistent and non persistent connections

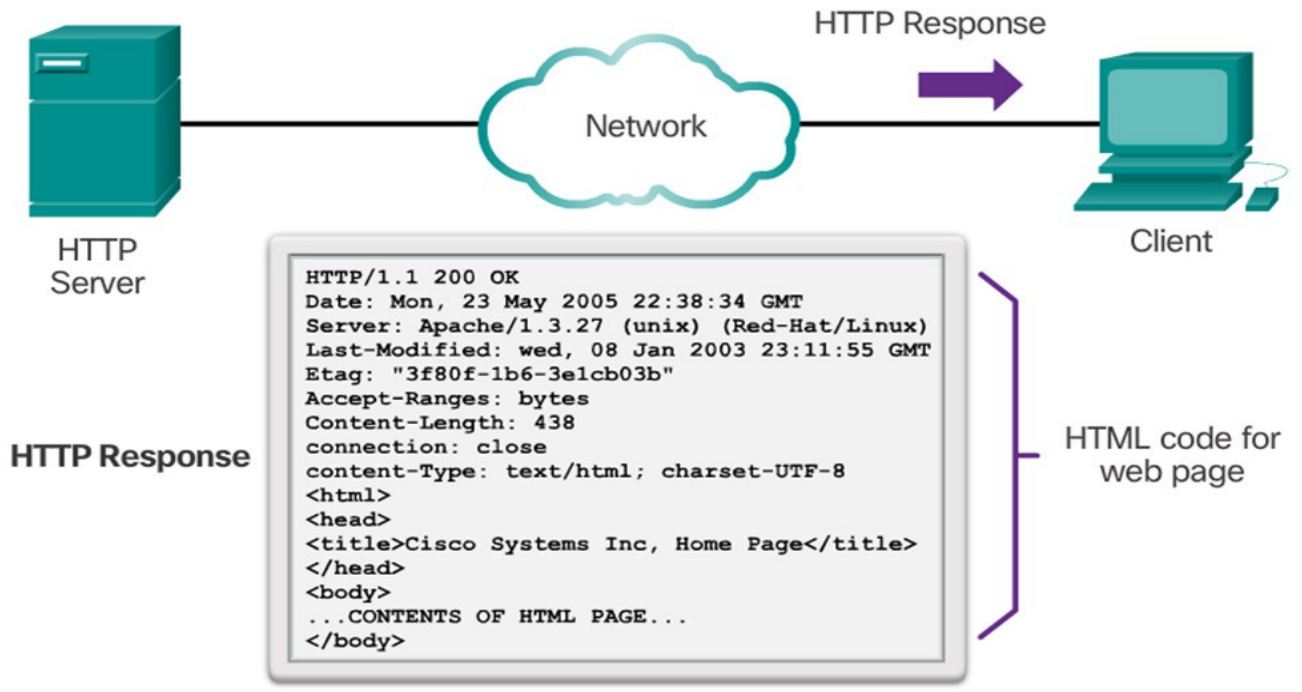
- HTTP, which can use both non-persistent connections and persistent connections. Although HTTP uses persistent connections in its default mode, HTTP clients and servers can be configured to use non-persistent connections instead.
- Non- persistent: each request/response sent over a separate connection.
- Persistent: All requests/responses sent over the same connection

HTTP & HTML

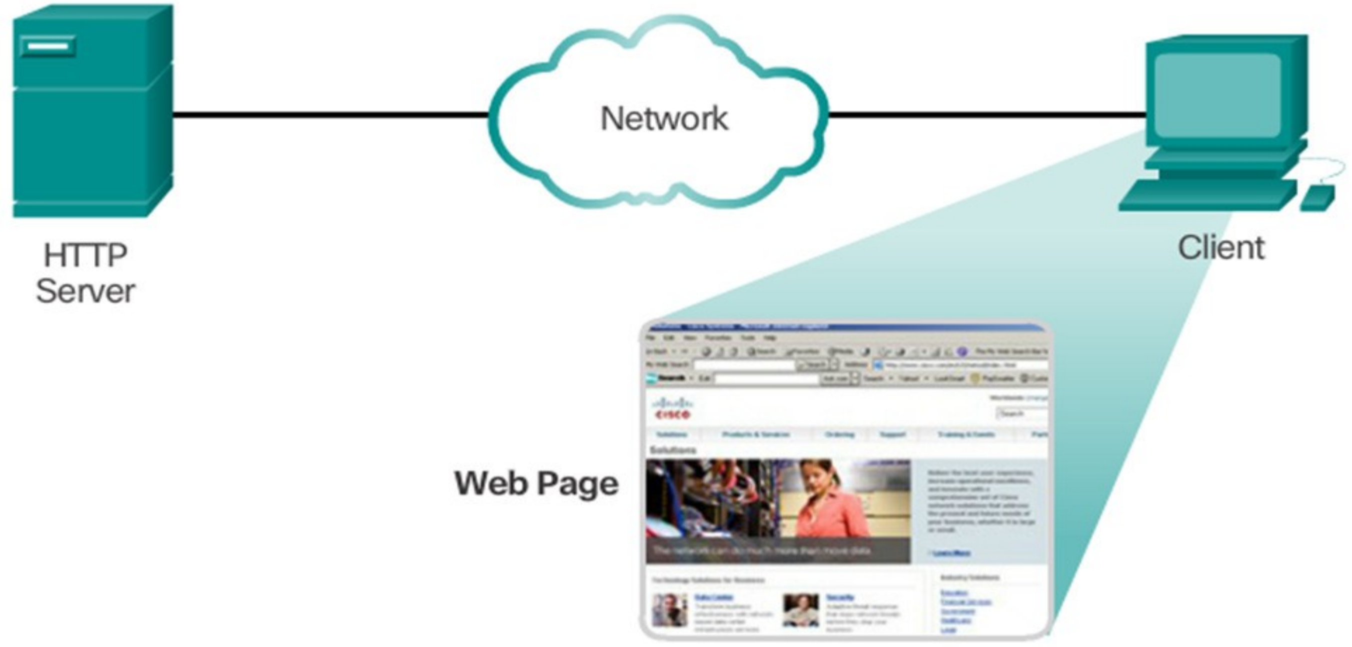
- A web address or uniform resource locator (URL) is a reference to a web server.
- A URL allows a web browser to establish a connection to that web server.
- URLs and Uniform Resource Identifier (URIs) are the names most people associate with web addresses.
- The URL http://cisco.com/index.html has three basic parts:
    - o **http** (the protocol or scheme)
    - o **www.cisco.com** (the server name)
    - o **index.html** (the specific filename requested)
- Using DNS, the server name portion of t URL is then translated to the associated address before the server can be contacted.



- The browser sends a GET request to the server’s IP address and asks for the **index.html** file.
- The server sends the requested file to the client.
- The **index.html** was specified in the URL and contains the HTML code for this web page.
- The browser processes the HTML code and formats the page for the browser window based on the code in the file.



```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.27 (unix) (Red-Hat/Linux)
Last-Modified: wed, 08 Jan 2003 23:11:55 GMT
Etag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
connection: close
content-Type: text/html; charset-UTF-8
<html>
<head>
<title>Cisco Systems Inc, Home Page</title>
</head>
<body>
...CONTENTS OF HTML PAGE...
</body>
```

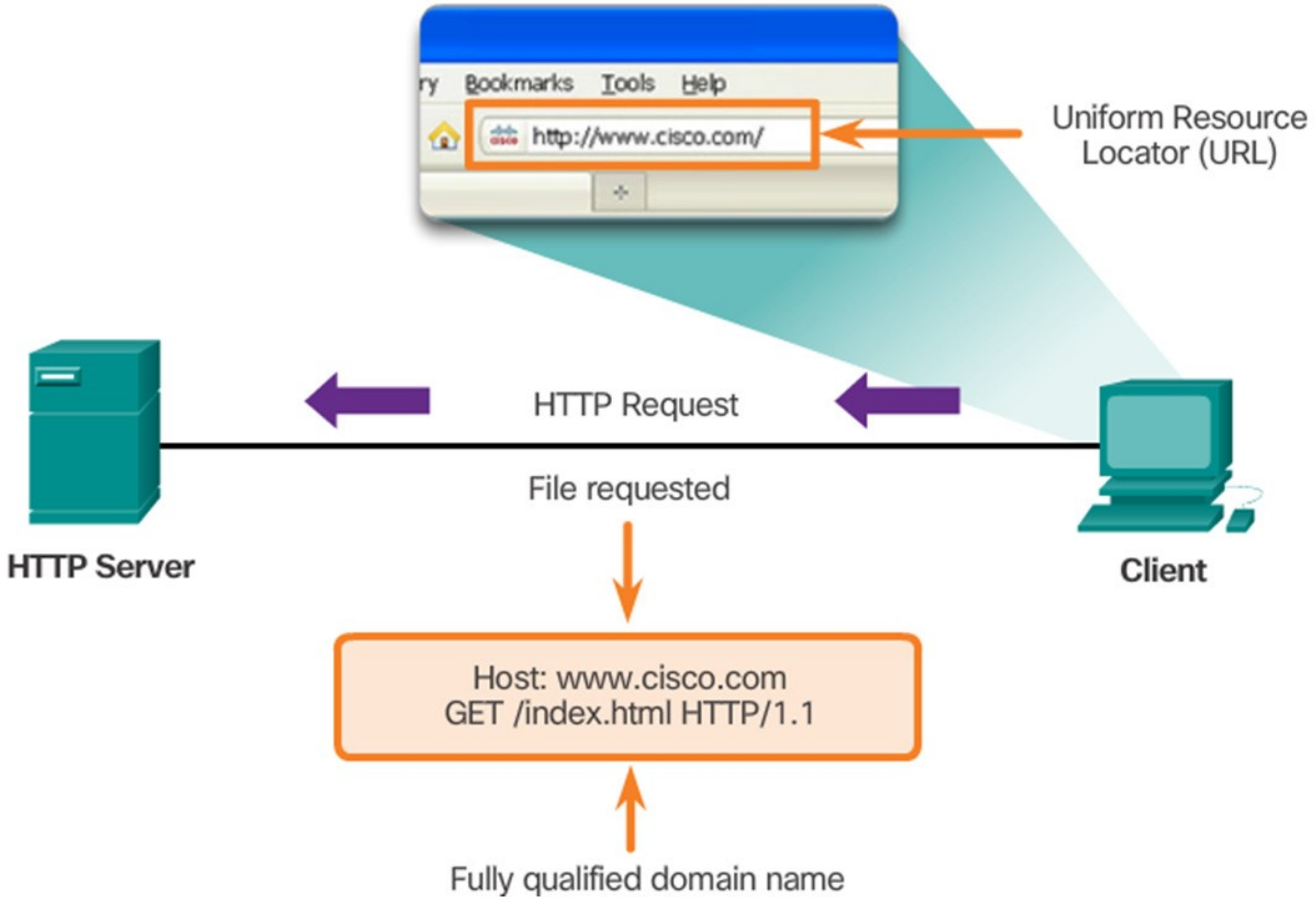


HTTP & HTTPS

- **HTTP**
  - o Is a request/response protocol.
  - o Has three common message types: GET, POST, PUT.
  - o Is not secure. Messages can be intercepted.
  - o port 80

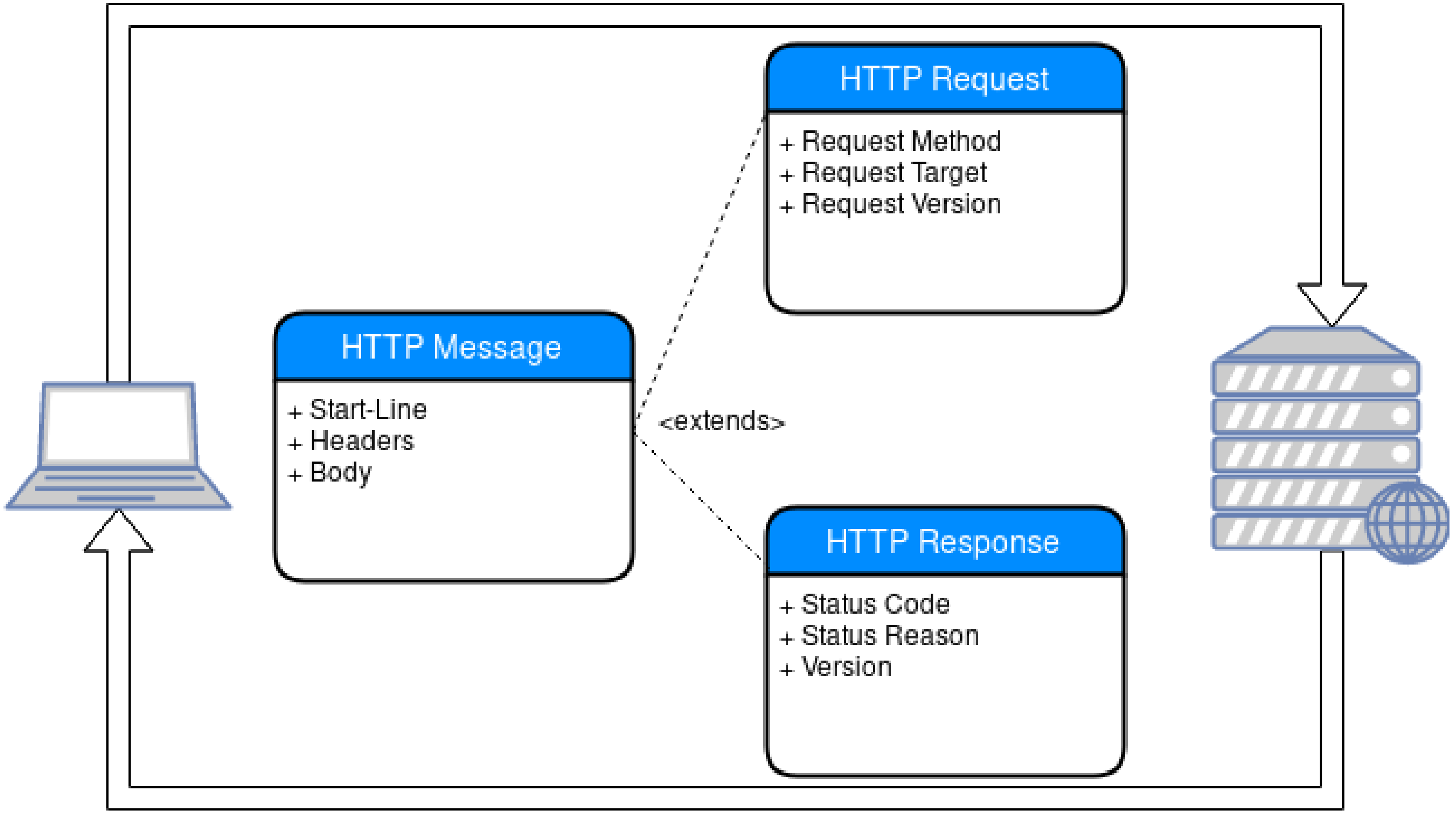
Protocol://servername:port/path

- **HTTPS**
  - o uses authentication and encryption to secure data.
  - o port 443



Structure of HTTP Message

Request Line	Status Line
General Header	
Request Header	Response Header
Entity Header	
Empty Line	
Message Body (entity body or encoded entity body	


HTTP Request
+ Request Method
+ Request Target
+ Request Version
HTTP Message
+ Start-Line
+ Headers
+ Body
<extends>
HTTP Response
+ Status Code
+ Status Reason
+ Version

HTTP Request Header

Below we provide a typical HTTP request message:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

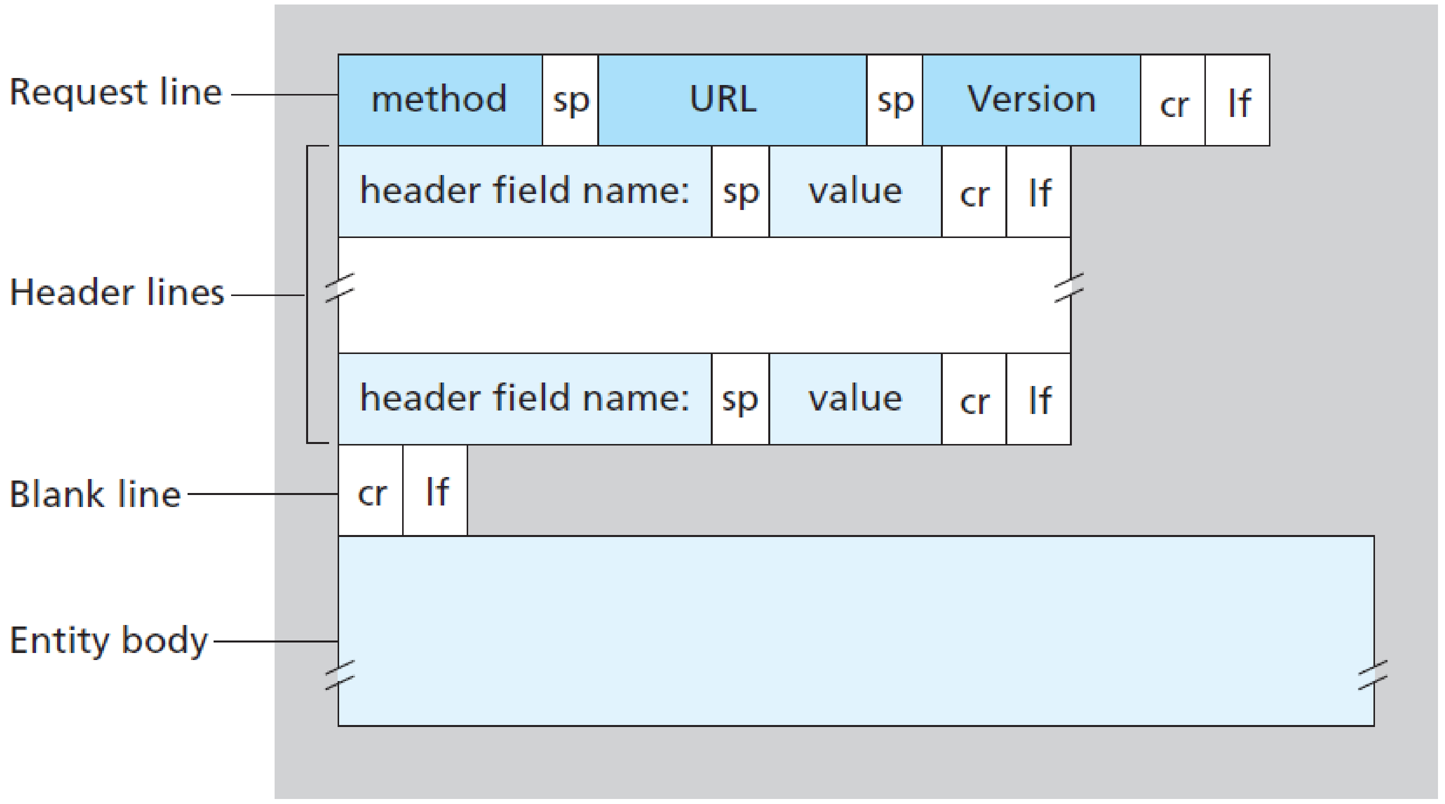


Figure 2.8 ♦ General format of an HTTP request message

HTTP Response Header

- HTTP/1.1 200 OK
- Connection: close
- Date: Tue, 09 Aug 2011 15:44:04 GMT
- Server: Apache/2.2.3 (CentOS)
- Last-Modified: Tue, 09 Aug 2011 15:11:03
- Content-Length: 6821
- Content-Type: text/html
- (data data data data data ...)

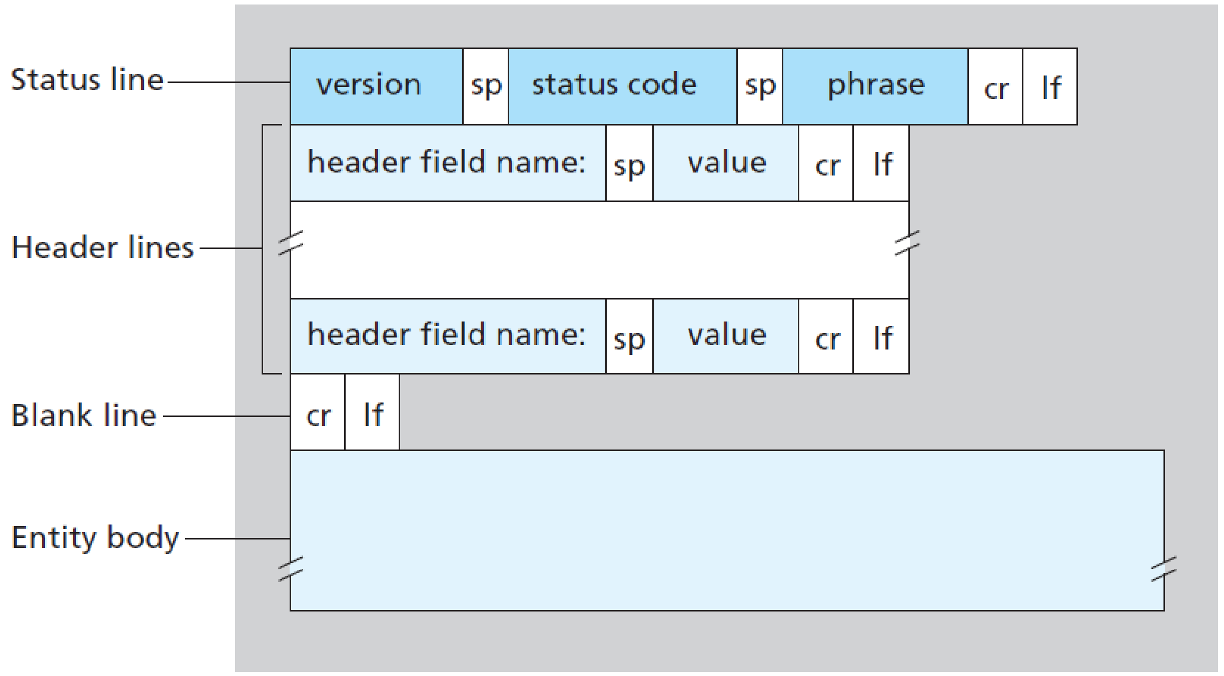


Figure 2.9 ♦ General format of an HTTP response message

References

- https://www.computernetworkingnotes.com/networking-tutorials/network-cable-types-and-specifications.html
- Data Communication and Computer Networks by Behrouz A Forouzan.pdf
- Data And Computer Communications by William Stallings.pdf
- https://www.academia.edu/37562829/INTRODUCTION_TO_COMPUTER_NETWORKS
- https://www.studytonight.com/computer-networks/osi-model-network-layer#:~:text=Network%20Layer%20%2D%20OSI%20Model,across%20multiple%20links%20(networks).&text=It%20routes%20the%20signal%20through,acts%20as%20a%20network%20controller