Chapter Objectives

• In this chapter, you will learn to:
  – Describe types of distributed software architecture and discuss their advantages compared with centralized applications
  – Explain how operating systems and network protocol stacks cooperate so that users and programs can access remote resources
  – Explain the role and function of directory services and the LDAP standard
  – Describe low-level protocols for interprocess communication across networks
Chapter Objectives (continued)

– Describe standard Internet protocols for accessing distributed resources
– Discuss component-based application development and describe the protocols and standards that support it
– Describe cloud computing models and compare their economic benefits and risks
FIGURE 13.1 Topics covered in this chapter
Courtesy of Course Technology/Cengage Learning

Internet and distributed application services
- Distributed computing
- Network resource access
- Directory services
- Interprocess communication
- The Internet
- Component-based application development
- Cloud computing

External resources
Network communication
Network software
Software
Hardware
Physical network
Distributed Software Architecture

• Distributes parts of an information system across many computer systems and locations
• Distributed computing or distributed processing
Client/Server Architecture

- **Server**
  - Manages system resources
  - Provides access to them through a well-defined communication interface

- **Client**
  - Uses communication interface to request resources
  - Server responds to those requests
FIGURE 13.2 Network printing services implemented with client/server architecture
Courtesy of Course Technology/Cengage Learning
N-Layer Client/Server Architecture

• Three-layer architecture
  – Divides application software into three tiers
    • Data layer
    • Business logic layer
    • View layer
  – Simplifies distributing or replicating application software across a network

• N-layer architectures
  – Employ more than three layers
FIGURE 13.3 Three-layer architecture
Courtesy of Course Technology/Cengage Learning
Middleware

• System software that implements communication standards and gives clients and servers the ability to interact

• More complex multitier application might need additional middleware
Peer-to-Peer Architecture

- Lower-level protocols
- Enable processes to communicate synchronously across a network
- Often used by system software to exchange data and coordinate activities
Network Resource Access

- OS components enable distributed access by:
  - Distinguishing between local and remote resources
  - Interacting with distant operating systems
Protocol Stacks

• Software that implements the lowest five levels of the OSI model
• Provide several advantages for implementing network I/O and services
FIGURE 13.4 Two protocol stacks with three shared layers
Courtesy of Course Technology/Cengage Learning

<table>
<thead>
<tr>
<th>Layer</th>
<th>Stack 1</th>
<th>Stack 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application layer</td>
<td>HTTP</td>
<td>Skype</td>
</tr>
<tr>
<td>Transport layer</td>
<td>TCP</td>
<td>UDP</td>
</tr>
<tr>
<td>Internet layer</td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td>Network Interface layer</td>
<td>Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>Physical layer</td>
<td>NIC &amp; Cat6 cable</td>
<td></td>
</tr>
</tbody>
</table>
Advantages of Protocol Stacks

• Divide network interaction into well-defined pieces that can be separately implemented, installed, and updated
• Provide flexibility to keep up with rapid protocol standard evolution
• Insulate application programs and portions of OS from details of low-level network communication protocols and physical network implementation (ensures software portability)
Static Resource Connection

- Initialized by user or system administrator prior to accessing a remote resource
- Difficult to initialize and maintain
FIGURE 13.5 Displaying and creating static connections in Windows 7
Courtesy of Course Technology/Cengage Learning
FIGURE 13.6 Software resources used to access local and remote resources
Courtesy of Course Technology/Cengage Learning
Dynamic Resource Connection

- Established through interaction between a resource layer and a primary resource registration repository
- More flexible but requires a distributed registry of resource name and locations
Directory Services

• Middleware that:
  – Stores name and network address of distributed resources
  – Responds to directory queries
  – Accepts directory updates
  – Synchronizes replicated or distributed directory copies

• Integral components of network operating systems
Information Stored in Network OS Directories

- Registered users and their permissions to access directory objects
- Shared hardware resources
- Shared files, databases, and programs
- Computer systems and specialized hardware devices
Lightweight Directory Access Protocol (LDAP)

• Widely deployed directory service standard that can track users, distributed resources, and objects

• Limited interoperability among different LDAP directories
  – Does not define standard content templates

• Defines several standard container types and an attribute called distinguished name (DN)
  – Which uniquely identifies the object within an objectclass
FIGURE 13.7 An LDAP hierarchy of objects and container objects
Courtesy of Course Technology/Cengage Learning
Microsoft Active Directory

- Directory service and security system built into Windows server
- Stores information about network resources
- Every resource or container object has an access control list that describes access rights
- Based on LDAP and the Internet Domain Naming Service (DNS)
- Does not support distributed or component-based software directly
FIGURE 13.8 Active Directory objects
Courtesy of Course Technology/Cengage Learning
FIGURE 13.9 Viewing an ACL
Courtesy of Course Technology/Cengage Learning
Interprocess Communication

• Distributed processes must communicate with one another to exchange data and synchronize activities
• Peer-to-peer interprocess communication protocols
  – Sockets
  – Named pipes
  – Remote procedure calls
  – Distributed Computing Environment (DCE)
FIGURE 13.10 Interprocess communication protocols layered over TCP/IP

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>DCE</td>
</tr>
<tr>
<td></td>
<td>RPC</td>
</tr>
<tr>
<td></td>
<td>Named pipes</td>
</tr>
<tr>
<td>Transport</td>
<td>Sockets</td>
</tr>
<tr>
<td>Internet</td>
<td>TCP</td>
</tr>
<tr>
<td></td>
<td>IP</td>
</tr>
</tbody>
</table>

Courtesy of Course Technology/Cengage Learning
Sockets

• Unique combinations of an IP number and a port number, separated by a colon (e.g., 129.24.8.1:53)

• Implement direct process-to-process communication via protocol stacks

• Supported by all modern OSs, which enable programs to initialize sockets, receive messages sent to a socket, and send messages to sockets anywhere on the Internet
FIGURE 13.11 Multiple processes communicating through sockets
Courtesy of Course Technology/Cengage Learning
Named Pipes

• Region of shared memory that enables multiple processes executing on the same machine to exchange data
• Name is permanently placed within file system directory
• Able to communicate among processes on different computers
FIGURE 13.12 Two processes communicating through a named pipe
Courtesy of Course Technology/Cengage Learning
Remote Procedure Calls

• Allow one process to execute another as a subroutine with parameter passing and format translation

• The calling process:
  – Passes parameters to the called process
  – Waits for the called process to complete its task
  – Accepts parameters back from the called process
  – Resumes execution with the instruction following the call
Distributed Computing Environment

• Combines all peer-to-peer approaches and adds security (Kerberos) and minimal directory services
• Promotes interoperability of distributed software across operating systems and middleware products
FIGURE 13.13 DCE software layers
Courtesy of Course Technology/Cengage Learning
# The Internet

| **Internet**       | • Global collection of networks interconnected using TCP/IP  
|                   | • Infrastructure upon which the Web is based |
| **WWW**           | • Collection of resources (programs, files, services), accessible over the Internet by standard protocols (FTP, HTTP)  
|                   | • Organized using client/server architecture |
| **Intranet**      | • Private network that uses Internet protocols; accessible only to a limited set of internal users  
|                   | • Set of privately accessible resources, organized and delivered via Web protocols over a TCP/IP network |
Standard Web Protocols and Services

• Web protocols
  – Define valid resource formats and a standard means of requesting resources
• Identified by a unique Uniform Resource Locator (URL)
FIGURE 13.14 URL components
Courtesy of Course Technology/Cengage Learning
<table>
<thead>
<tr>
<th>Category</th>
<th>Sample protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formatted and hyperlinked</td>
<td>Hypertext Markup Language (HTML) and Extensible Markup Language (XML)</td>
</tr>
<tr>
<td>documents</td>
<td></td>
</tr>
<tr>
<td>File and document transfer</td>
<td>File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP)</td>
</tr>
<tr>
<td>E-mail and messaging</td>
<td>Simple Mail Transfer Protocol (SMTP) and Internet Message Access Protocol (IMAP)</td>
</tr>
<tr>
<td>Videoconferencing</td>
<td>H.323 and Session Initiation Protocol (SIP)</td>
</tr>
<tr>
<td>Executable programs</td>
<td>Java, JavaScript, and VBScript</td>
</tr>
<tr>
<td>Interprocess communication</td>
<td>Remote Procedure Call (RPC) and named pipes</td>
</tr>
<tr>
<td>Web services</td>
<td>Simple Object Access Protocol (SOAP) and Universal Description, Discovery, and Integration (UDDI)</td>
</tr>
<tr>
<td>Instant messaging</td>
<td>SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE), Instant Messaging and Presence Service (IMPS), and Extensible Messaging and Presence Protocol (XMPP)</td>
</tr>
</tbody>
</table>

**TABLE 13.1 Web protocols**
FIGURE 13.15 Telnet connection
Courtesy of Course Technology/Cengage Learning
The Internet as an Application Platform

- Application program executes on a Web server that can be accessed from any computer with an Internet connection
FIGURE 13.18 A distributed Web-based application
Courtesy of Course Technology/Cengage Learning
The Internet as an Application Platform

<table>
<thead>
<tr>
<th><strong>Advantages</strong></th>
<th><strong>Disadvantages</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Expands accessibility</td>
<td>• Security</td>
</tr>
<tr>
<td>• Eliminates need to install custom client software</td>
<td>• Performance</td>
</tr>
<tr>
<td>• Cheaper to develop and deploy; built around existing Web standards and browser software already installed on clients</td>
<td>• Reliability</td>
</tr>
</tbody>
</table>
Components and Distributed Objects

• Component
  – Standardized, interchangeable software module that is executable, has a unique identifier, and has a well-known interface

• Distributed object
  – Cooperating process that implements a public interface to the services it provides
Component-Based Software

- Enables construction of complex programs and applications from smaller previously developed parts
- Requires protocols and infrastructure for component registration, discovery, and communication
Components and Objects

• Send and respond to messages
• Encapsulate internal data
• Interact with other components through a well-defined interface
Connection Standards and Infrastructure

• Interoperability requires well-defined and widely adopted standards

• Standard network protocols do not address:
  – Format and content of valid messages and responses
  – Way to uniquely identify each component on the Internet and route messages to/from that computer

• Additional standards provided by CORBA, COM+, SOAP, J2EE
Common Object Request Broker Architecture (CORBA)

• Specifies middleware used by objects to interact across networks

• Key components
  – Object Request Broker (ORB)
  – Internet Inter-ORB Protocol (IIOP)

• Robust, scalable, independent of programming language, OS, and CPU architecture

• Disadvantage: complexity
Component Object Model Plus (COM+)

• Like CORBA, defines component registration, message routing services, and component communication protocol
• Unlike CORBA, components are not assigned permanent identifier, and are registered in Windows Registry of client machine where installed
• Disadvantage: dependence on propriety technology and limited support outside of Microsoft products
Simple Object Access Protocol (SOAP)

- Attempts to address shortcomings of CORBA and COM+
- Has few infrastructure requirements and relatively simple programming interface
- Relies on existing Internet protocols
- Disadvantage: security and message delivery guarantees
FIGURE 13.20 Client/server communication with a SOAP message
Courtesy of Course Technology/Cengage Learning
Java 2 Enterprise Edition (J2EE)

- Standards for developing/deploying component-based distributed applications written in Java
- Key elements follow three-layer architecture
- Component interactions based on many standards
  - Remote Method Invocation (RMI)
  - Java Naming and Directory Interface (JNDI)
  - Java Authentication and Authorization Service (JAAS)
  - Java Database Connectivity (JDBC)
FIGURE 13.21 Java EE architecture
Courtesy of Course Technology/Cengage Learning
Emerging Distribution Models

• Coarse-grained distribution models to allow businesses to streamline operations and costs
• Models emerged in the early 2000’s
• Initial uses were for business-to-business processes
• Made possible by high-speed Internet access for many businesses
• Internet compared to electrical grid, but electricity is uniform and technology services are not
FIGURE 13.22 Cloud distribution models
Courtesy of Course Technology/Cengage Learning
Software as a Service (SaaS)

• Web-based architectural approach
  – Users interact via a Web browser or other Web-enabled view layer with application software provided by a third party
• Eliminates the need to purchase hardware and install software
• Service provider upgrades and maintains the application for its customers
Platform as a Service (PaaS)

• Architectural approach
  – An organization rents access to system software and hardware on which it installs its own application software and other services
• Simplest example is a hosted Web site
• Complex examples can include complete e-commerce sites
• Main advantage is avoiding the need to operate its own servers and system software
Infrastructure as a Service (IaaS)

- Similar in many ways to PaaS, but the service provider supplies little or no system software
- Hardware virtualization is a key supporting technology
- Service provider delivers a generic platform for virtual servers
- Examples include Google or Amazon’s storage services
Risks

• Vendor reliability is a critical issue
• Vendor lock-in is a risk, although its level varies across architectures
• Service availability
• Data security and privacy
• Legal ownership of stored data
FIGURE 13.23 Vendor lock-in risk is lowest for IaaS and highest for SaaS
Courtesy of Course Technology/Cengage Learning
Summary

- Distributed computing
- Network resource access
- Interprocess communication
- The Internet
- Components and distributed objects
- Distributed computing