CNT 4714: Enterprise Computing Spring 2011

Java Networking and the Internet – Part 1

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cnt4714/spr2011

Department of Electrical Engineering and Computer Science
University of Central Florida

Distributed Applications in the Enterprise

- Distributed applications are one of the latest developments of information technology, which began about 50 years ago, and is still developing at a very fast pace.
- The first electronic computers available in 1940s and 50s were reserved for special applications. Many had military applications such as the encryption and decoding of messages.
- The 1960s witnessed the advent of batch processing, in which several users could pass their tasks to the computer operator (the "server"). Once processed, the results were returned to the "client" by the operator. As there was no interactivity at that time, computers were used for primarily numerical applications that required little user input but required a high computational effort.

Distributed Applications in the Enterprise (cont.)

- With the advent of mainframes, interactive applications came into play. Several users could finally use one computer simultaneously in time-sharing mode. Tasks were no longer completed in sequence as with batch processing, but rather completed in sections.

- The next trend, beginning with the introduction of the PC in 1980, was the shift in computing power from the central mainframe to the desktop. Computer performance at levels which were undreamt of previously, was now available to employees directly at their desk. Each user could install their own applications to create an optimally configured work environment. This began the age of the standardized office packages, which enabled office automation to be driven forward considerably.

Distributed Applications in the Enterprise (cont.)

- Since the 1990s, the trend has shifted increasingly from distributed information processing to enterprise computing.
- Previously autonomously operating workstations were integrated together with central file, database, and application servers, resulting in huge decentralized clusters, which were used to handle tasks of a more complicated nature.
- The defining sentence which characterized this phase coined by Sun Microsystems reads – "The network is the computer."
- What was it that led to this ever increasing greater importance of distributed applications?
- There are several reasons commonly cited:

Distributed Applications in the Enterprise (cont.)

1. The cost of chip manufacturing dropped sharply, enabling cheap mass production of computers.

2. Simultaneously, network technologies were developed with higher bandwidths – a necessity for the quick transfer of large amounts of data between several computers.

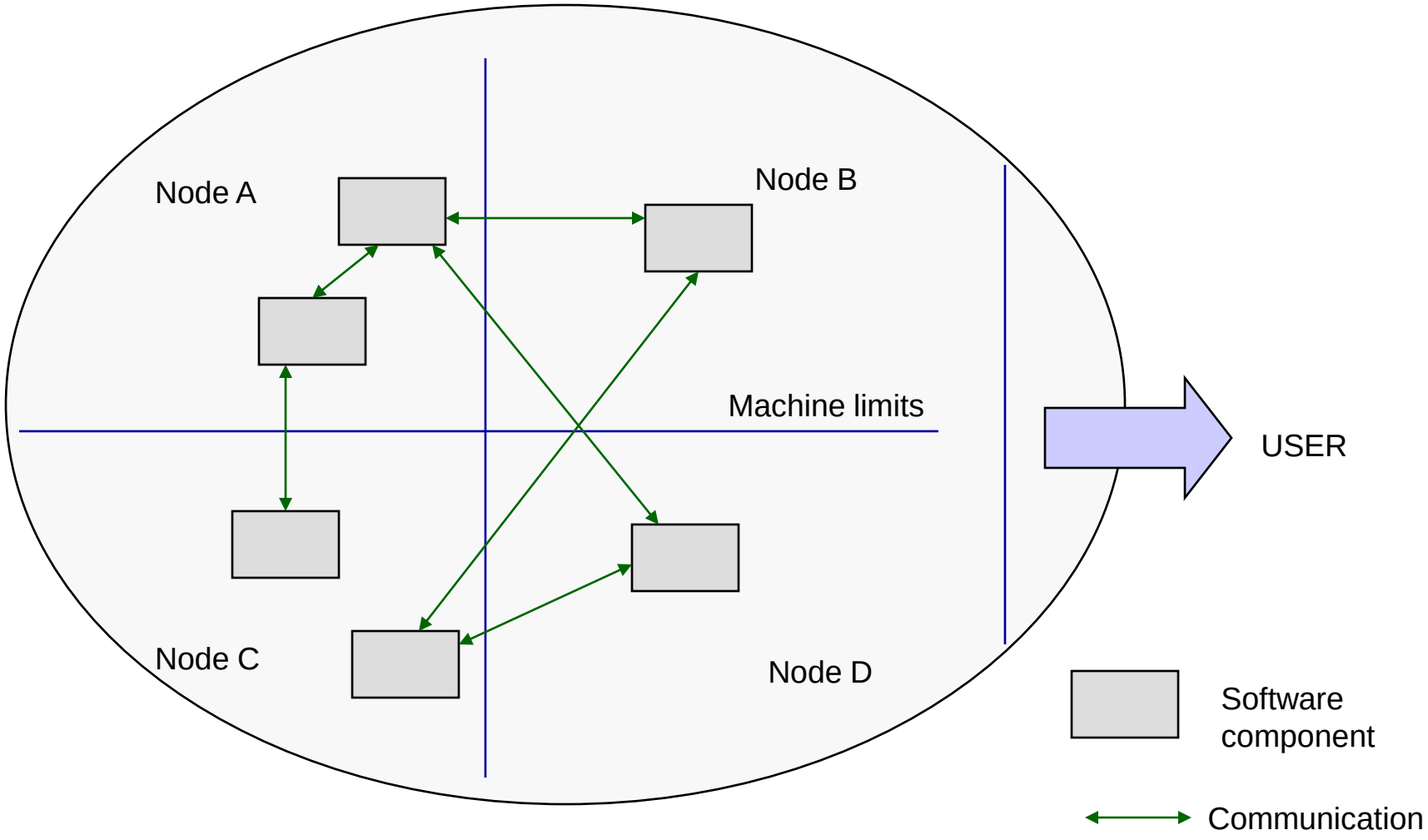
3. Response times became increasingly longer due to the heavy use of large mainframes, resulting in excessive waiting times.

4. The availability of a comparable distributed work environment gave rise to the desire for new applications that were not possible in a centralized environment. This development led from the first e-mail applications via the WWW to common use of information by people in completely different places.

What is a Distributed System?

- While many different definitions of what constitutes a distributed system have been put forth, there is general consensus that there are several central components that a distributed system must contain:
  - A set of autonomous computers.
  - A communication network, connecting those computers.
  - Software which integrates these components with a communication system.

What is a Distributed System?



What is a Distributed Application?

- A distributed application is an application A, the functionality of which is subdivided into a set of cooperating subcomponents A_1, A_2, ..., A_n with $n > 1$. The subcomponents A_i are autonomous processing units which can run on different computers and exchange information over the network controlled by coordination software.
- There are typically three levels defined for a distributed system:

Level 3	Distributed Applications
Level 2	Coordination Software
Level 1	Distributed Computer System

What is a Distributed Application? (cont.)

- The application on level 3 will ideally "know" nothing of the distribution of the system, as it uses the services of level 2, the administration software that takes over the coordination of all the components and hides the complexity from the application.
- In turn, level 2 itself uses the available distributed computing environment.

As an aside, a more humorous definition of a distributed system was given by Leslie Lamport (the guy who developed LaTex), who defined a distributed system as a system "in which my work is affected by the failure of components, of which I knew nothing previously."

Important Characteristics of Distributed Systems

- Based on our simple definition, there are several important characteristics of distributed systems that need a closer look.
- All of these characteristics are based on the concept of transparency.
- In the context of information technology, the concept of transparency literally means that certain things should be invisible to the user. The manner in which the problem is solved is largely irrelevant to the user.
- The following transparency properties play a large role in achieving this result for the user:

Transparency Properties of Distributed Systems

Location Transparency – users do not necessarily need to know where exactly within the system a resource is located which they wish to utilize. Resources are typically identified by name, which has no bearing on their location.

Access Transparency – the way in which a resource is access is uniform for all resources. For example, in a distributed database system consisting of several databases of different technologies, there should also be a common user interface (such as SQL).

Replication Transparency – the face that there may be several copies of a resource is not disclosed to the user. The user has no need to know whether they are accessing the original or the copy. The altering of the resource also must occur transparently.

Transparency Properties of Distributed Systems
(cont.)

Error Transparency – users will not necessarily be informed of all errors occurring in the system. Some errors may be irrelevant, and others may well be masked, as in the case of replication.

Concurrency Transparency – distributed systems are usually used by several users simultaneously. It often happens that two or more users access the same resource at the same time, such as a database table, printer, or file. Concurrency transparency ensures that simultaneous access is feasible without mutual interference or incorrect results.

Migration Transparency – using this form of transparency, resources can be moved over the network without the user noticing. A typical example is today's mobile telephone network in which the device can be moved around freely, without any loss of communication when leaving the vicinity of a sender station.

Transparency Properties of Distributed Systems (cont.)

Process Transparency – It is irrelevant on which computer a certain task (process) is executed, provided it is guaranteed that the results are the same. This form of transparency is an important prerequisite for the successful implementation of a balanced workload between computers.

Performance Transparency – when increasing the system load, a dynamic reconfiguration may well be required. This measure for performance optimization should be unnoticed by other users.

Scaling Transparency – if a system is to be expanded so as to incorporate more computers or applications, this should be feasible without modifying the system structure or application algorithms.

Language Transparency – the programming language in which the individual subcomponents of the distributed system or application were created must not play any role in the ensemble. This is a fairly new requirement of distributed systems and is only supported by more recently developed systems.

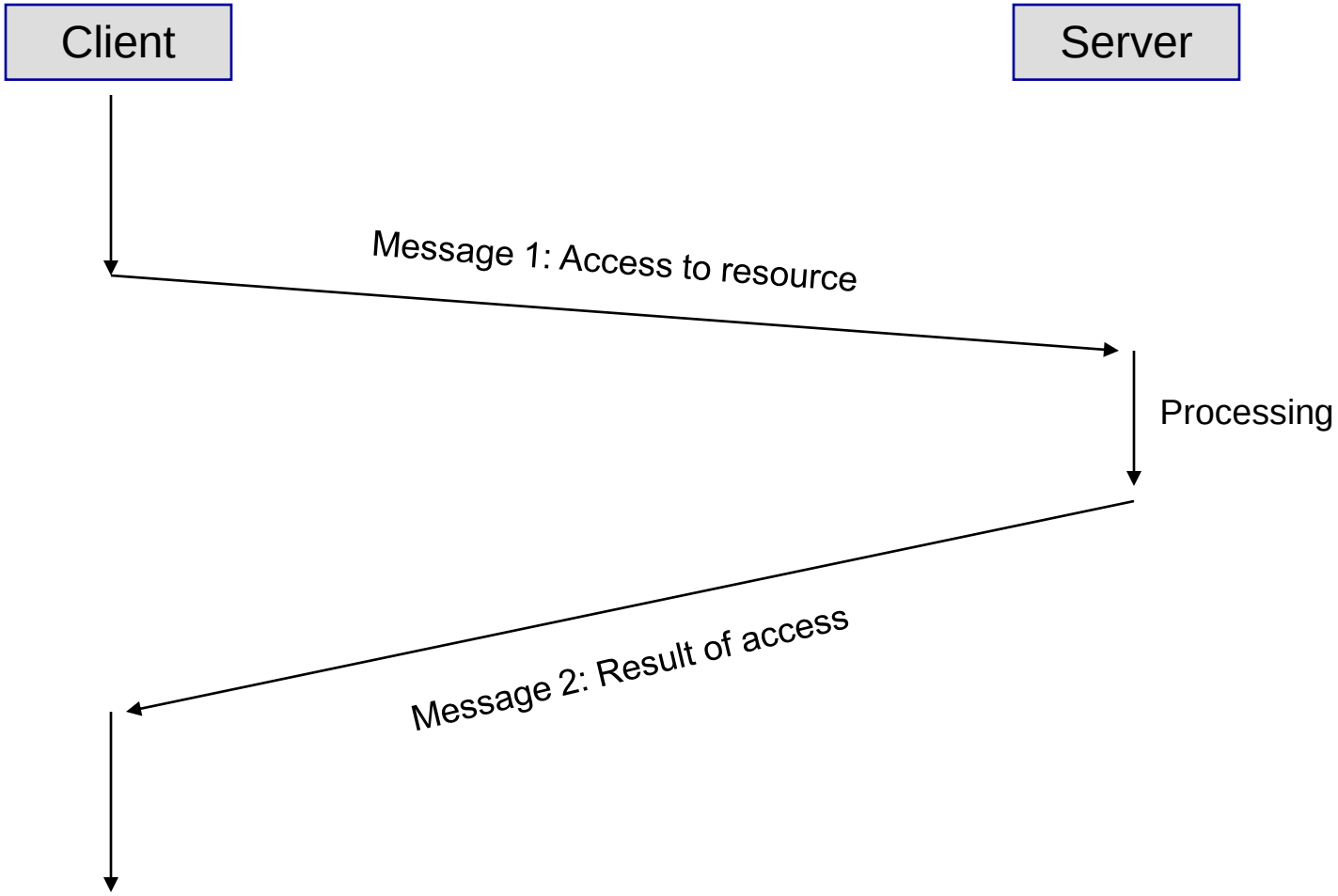
Basic Communication Models

- Communication between the individual components of a distributed system can occur in two basic ways: using either shared memory or message passing.
- Shared memory is an indirect form of communication, as both partners exchanging information do not communicate directly with each other, but via a third component: the shared memory.
- Message passing is a direct form of communication between the sender and receiver by means of a communication medium. Two functions are generally available for the execution of message exchange, usually called send and receive.

Basic Communication Models (cont.)

- Send is defined as: `send(r: recevier, m: message)`
  - This function sends the message *m* to the receiver *r*.
- Receive is defined as: `receive(s: sender, b: buffer)`
  - This function waits for a message from sender *s* and writes it in buffer *b* (part of the memory made available for the application process).
- The basic form of exchange of a single message can be combined with more complex models. One of the most important of these models is the client-server model.
  - In this model, the communication partners adopt the role of either a client or a server. A server is assigned to administer access to a certain resource, while a client wishes to use the resource.

Message Exchange in the Client-Server Model



Advantages and Disadvantages of Distributed Systems

- When compared to the mainframe approach, distributed systems offer the following advantages:
  - More economical – greater computing power is available at a lower cost.
  - Response times are much shorter.
  - Provide a better model of reality than a centralized computer (consider the information infrastructure of a multinational corp.).
  - Distributed systems can be made more reliable than a central system. Availability of individual components can be enhanced through replication. Also, the failure of a non-replicated component does not typically lead to total system failure as with a mainframe.
  - Distributed systems can be extended and adapted to increasing requirements far easier than can a mainframe.

Advantages and Disadvantages of Distributed Systems (cont.)

- When compared to the conventional PC approach, distributed systems offer the following advantages:
  - Generally speaking, communication between computers can only occur using a connection. Applications such as e-mail are only possible using this approach.
  - Networking PCs allows common use of both resources and data, especially hardware resources such as printers and hard drives.
  - Unless the PCs are networked, load sharing is not possible, so one user running two computationally-intensive applications will suffer even if the adjacent workstation is unused.

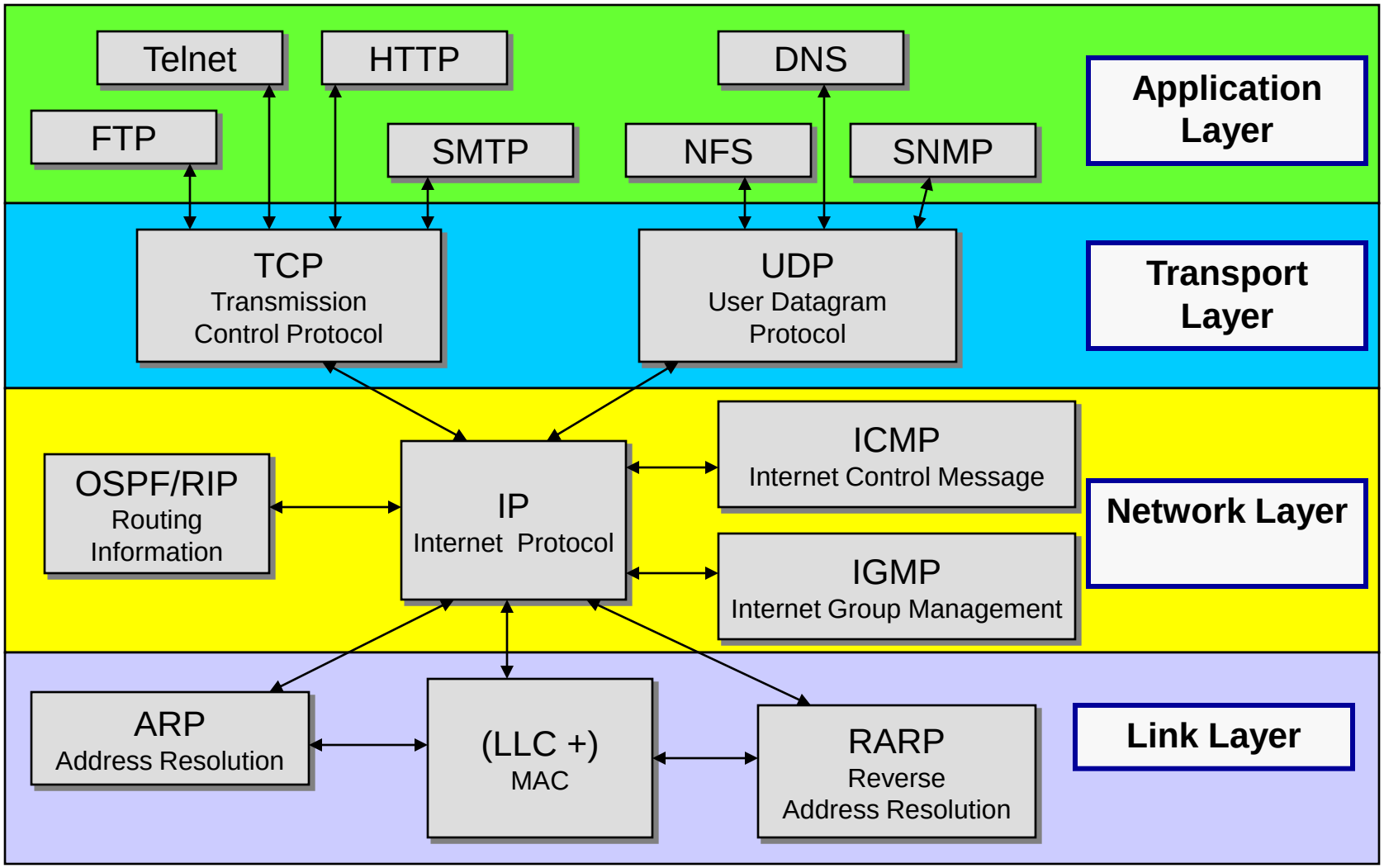
Advantages and Disadvantages of Distributed Systems (cont.)

- There are however, a few problems that arise with distributed systems:
  - The entire system is extremely dependent on transmission performance and the reliability of the underlying communication network. If the network is constantly overloaded, then the advantages of a distributed system are very quickly cancelled out, particularly with respect to response times.
  - Distribution and communication are always an increased security risk in many ways. Communications can be "snooped". Physical security of the system components becomes more difficult. Software issues concerning modification and piracy become more prevalent.
  - Software for both applications and the coordination of application components becomes more complex leading to greater chance for errors and higher development costs.

Technical Principles of the Internet

- Communications systems such as the Internet are best described using layered models because of their complexity.
- Every layer within the model has a certain task, and all layers together produce a particular communication service for the user.
- The layers are arranged in hierarchical form. Layers lower in the hierarchy produce a service used by the higher layers. The uppermost layer finally combines all lower layer services and constitutes the interface for applications.
- For the Internet, the so-called Internet reference model is used and is shown on the next slide.

Internet Reference Model



Internet Reference Model (cont.)

- The Link Layer describes the possible sub-networks of the Internet and their medium access protocols. These are, for example, Ethernets, token rings, FDDI, or ISDN networks. To its upper layer, the link layer offers communication between two computers in the same sub-network as a service.
- The Network Layer unites all the sub-networks to become the Internet. The service offered involves making communication possible between any two computers on the Internet. The network layer accesses the services of the link layer, in that a connection between two computers in different networks is put together for many small connections in the same network.

Internet Reference Model (cont.)

- The Transport Layer oversees the connection of two (or more) processes between computers communicating with each other via the network layer.
- The Application Layer makes application-specific services available for inter-process communication. These standardized services include e-mail, file transfer and the World Wide Web.
- Within the layers, protocols are used for the production of a service. Protocols are instances which can be implements either in hardware or software, and communicate with their partner instances in the same levels, but on other computers. It is only this cooperation that enables the service to be produced for the next level up.

Internet Reference Model (cont.)

- The TCP/IP Protocol constitutes the core of Internet communication technology in the transport and network layers.
- Every computer on the Internet always has an implementation of both protocols, TCP (Transmission Control Protocol) and IP (Internet Protocol).
- The task of IP is to transfer data from one Internet computer (the sender) to another (the receiver). On this basis, TCP then organizes the communication between the two processes on these two computers.

Some Important Application Layer Internet Protocols

- Telnet – makes a terminal emulation available on the remote computer. The protocol enables logins to other computers using the network.
- HTTP – (Hypertext Transport Protocol) is the underlying protocol of the World Wide Web. It is responsible for the transfer of hypertext documents.
- SMTP – (Simple Mail Transfer Protocol) is the protocol used for the transfer of e-mail messages.
- FTP – (File Transfer Protocol) is able to manage filestores on a server and enables clients to access files.
- SNMP – (Simple Network Management Protocol) is used for network management on the Internet.
- DNS – (Domain Name Service) is responsible for the mapping of symbolic names to IP addresses.
- NFS – (Network File System) makes the basic functionality for a distributed file system available.

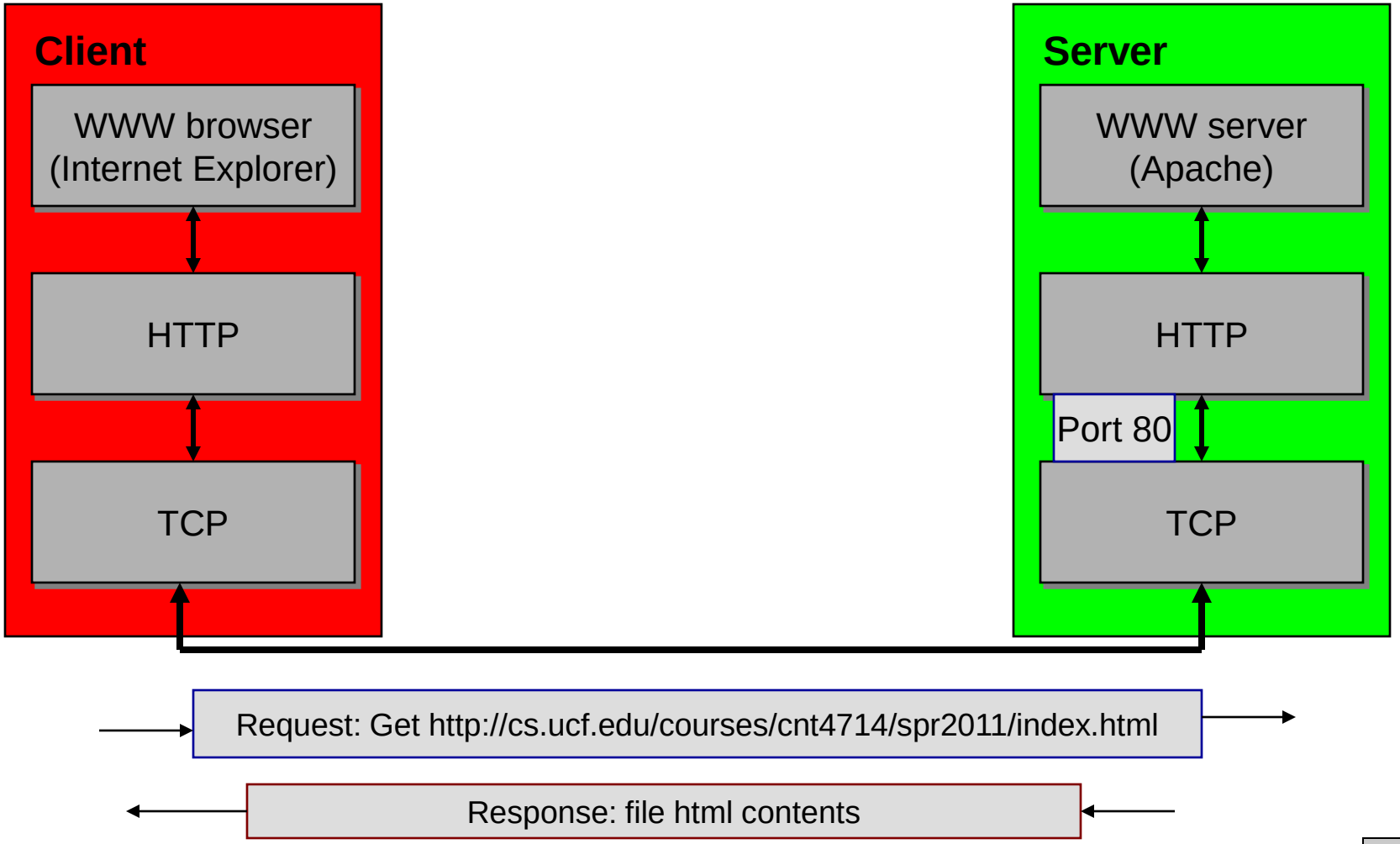
Basic Constituents of Web Applications

- In order to access the Web, first a web server is required. The server administers the entire data material intended for publication on the Web.
- The web server is also responsible for replying to client requests, by delivering the desired documents according to the entitlement of the client.
- Web servers usually record all Web files access, so different analyses can be made using the log files created, from the simplest of tasks such as how many hits have been made in a certain time period, or an analysis of the geographical distribution of users, to more sophisticated tasks such as monitoring attempts at unauthorized access.
- Web servers might also start other programs executing, with which additional information can be obtained or generated. It is this capability that forms the basis of all distributed applications on the WWW.

Basic Constituents of Web Applications (cont.)

- The communication between client and server on the WWW takes place using the HTTP protocol.
- HTTP is a purely text-based protocol. This means that the requests for a document are transferred by the client to the server using a "readable" command such as "get". The server responds to the request by making the requested document available to the client, together with a header giving further information. The server may sometimes respond with an error message, such as if the requested document is not available or the user does not have the proper permission to view the document.
- This protocol is illustrated on the next page. HTTP uses the TCP service for the actual transfer of data between client and server. For every transfer of a Web document, a TCP connection is first established, via which HTTP protocol messages are transferred. (Actually, the TCP connection persists over several HTTP requests.)

The Architecture of a Web Service



Construction of Web Applications

- The simplest form of an application on the WWW is that in which the provider places a number of static documents on a server.
- A static document is a document which can only be changed from outside, by human intervention.
- Clearly, this prototype is not flexible. As soon as information has to be modified on the server, a slow and cost-intensive process is required. For many applications, this process is simply not an option. Consider, for example, a provider that publishes current weather information. Since this information is constantly changing, an employee would need to be constantly updating the web pages.
- There are a number of approaches today which allow for the dynamic creation of web pages, some of which are already fairly old and others are relatively new. Among the newer of these are Java Servlets and Java Server Pages that we will see later in the semester.

The Architecture of Distributed Web Applications

- There are basically four major components which can or could constitute a distributed application on the Internet. These are:

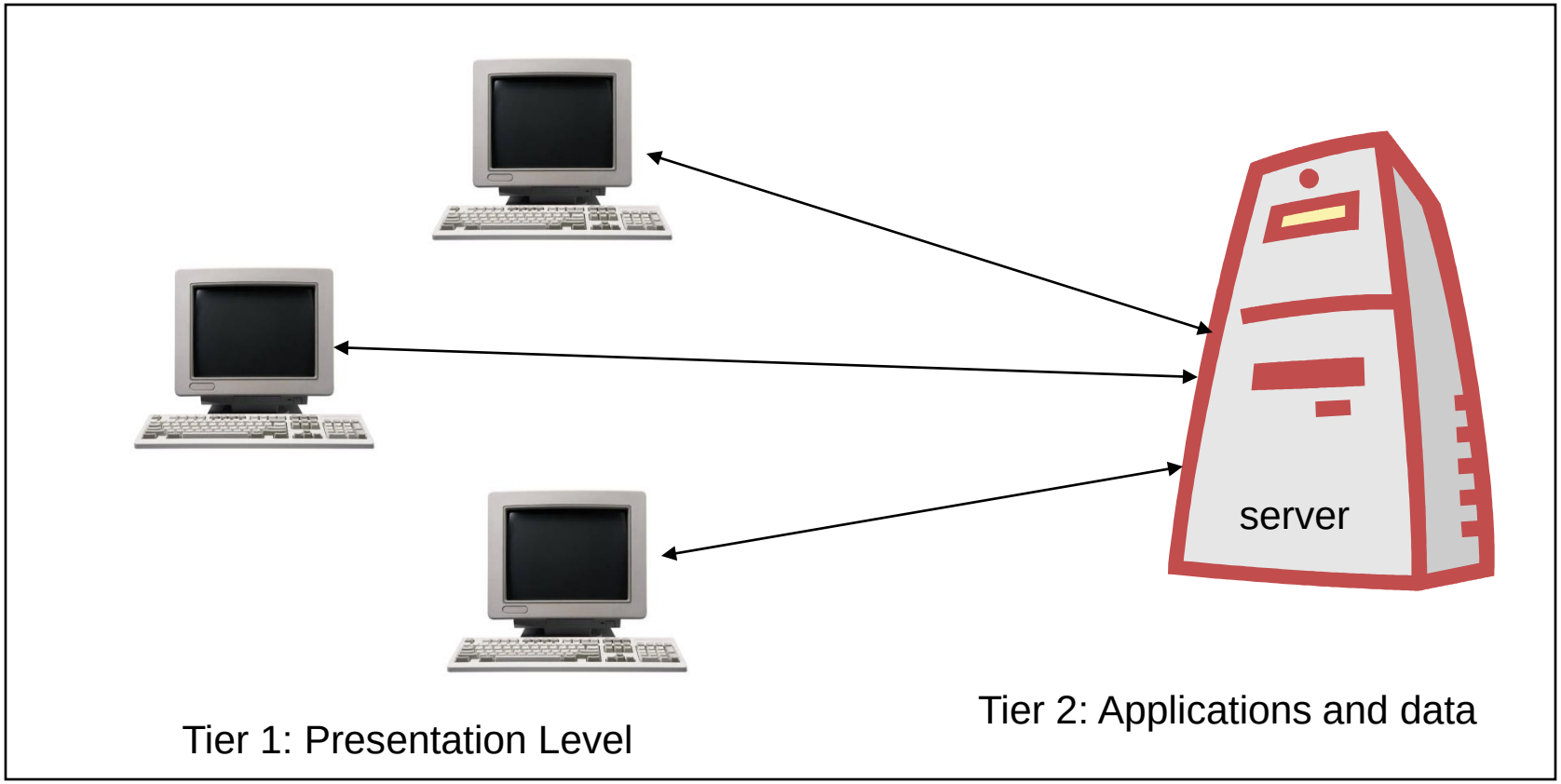
  1. The presentation interface to the user, as well as access programs to server components.
  2. An access interface to server components.
  3. The server application logic.
  4. File storage, databases, etc.

- In distributed applications, these four generic components can be distributed on the physical nodes of the system in different configurations.

- The term n-tier architecture was coined for the different variants that can be produced. The term indicates the number of levels on which the components are distributed. In practice, 2-, 3-, and 4-tier architectures are used.

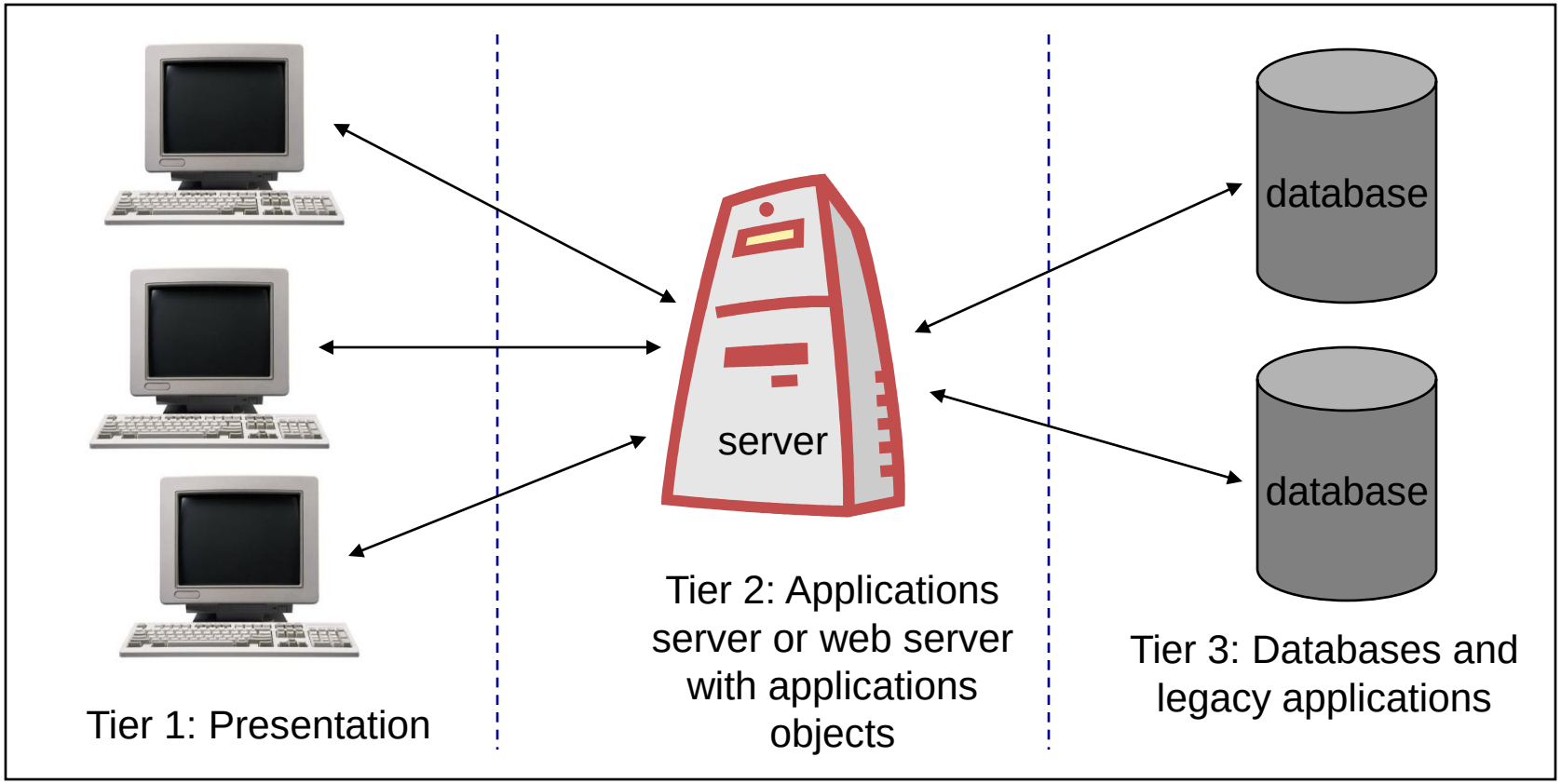
A Two-Tier Architecture

- The simplest version is the 2-tier architecture in which the presentation components are placed on the client computers, and all other components reside on one server computer. The most common example of this is TCP based client-server applications in which databases are accessed directly from the server process.



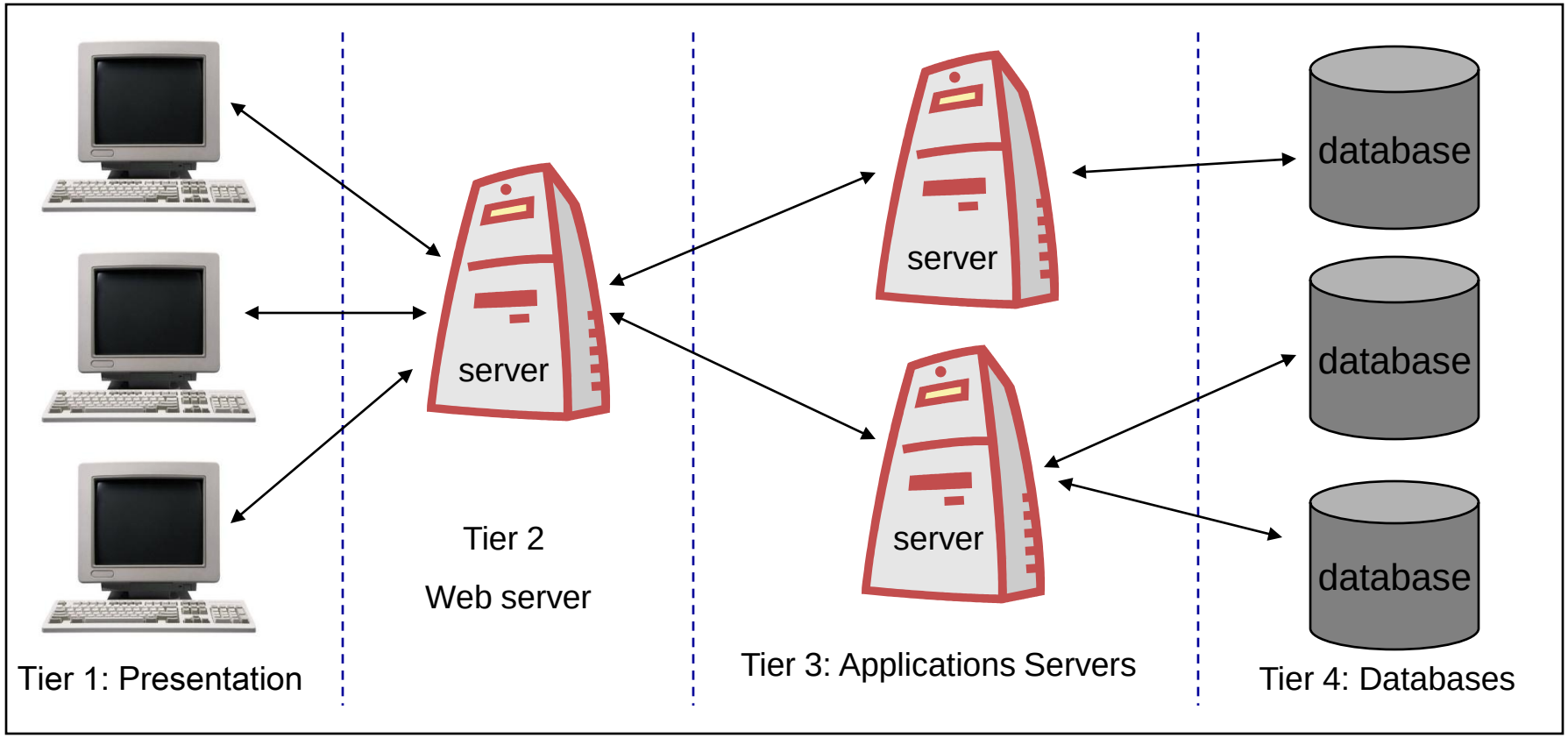
A Three-Tier Architecture

- The 3-tier architecture goes one step further, so that the actual applications are separated from data stocks. This is the common configuration for most servlet applications.



A Four-Tier Architecture

- The 4-tier architecture refines the 3-tier version by partitioning the server interface from the applications. Although not as common as the 3-tier version, this is the common configuration for many CORBA applications.



Thin Clients

- The extensive partitioning of the architecture of a distributed system into different components with respectively different areas of responsibility, basically allows for the creation of simpler and therefore more controllable individual components.
- The result of this on the client side is the development of thin clients. A thin client is a client program which contains almost no application logic, but offers only the presentation interface to the actual application program, which may run in a distributed fashion on several servers.
- While the application is executed and the graphical interface is in use, application logic is partly loaded on the client computer and executed there locally. However, it is not loaded from the local hard drive, but always by a server via the network.
- The most common version of a thin client today is a web browser. A web browser has no information whatsoever on specific applications.

Thin Clients (cont.)

- A web browser is only able to represent web pages, and possibly execute applets.
- If a certain application is to be used, then the corresponding web pages must be loaded by a web server.
- The use of thin clients has several advantages (as opposed to heavy clients):
  - The installation of program components on the client computer is unnecessary. Neither a reconfiguration of the computer nor regular updates of client software are required.
  - Users do not have to adjust to a new user interface for every distributed application. Access is always made using a well-known web browser interface. This renders a potentially large amount of training unnecessary.
  - Client computers can, on the whole, be equipped more inexpensively, as large hard drives for storing application programs are not needed.