Distributed Computing Systems

Overview

Outline

Overview
Network Infrastructure
Distributed Computing Systems
Communication Protocols
Communication Models
Communication Subsystems
Distributed Applications
Summary

Overview

- *Observation*
  - Stand-alone computers are increasingly being interconnected to form Distributed Computing Systems (DCS)
- *Evolution*
  - "Old" days: a computer was a stand-alone machine
    - ▷ *e.g.*, mainframes, PC
  - Today: computers communicate with each other
    - ▷ *e.g.*, "the network is the computer"
- *Key themes*:
  - Open systems and international standards...

Overview (cont'd)

- Three phases mark the evolution of networking and distributed systems:

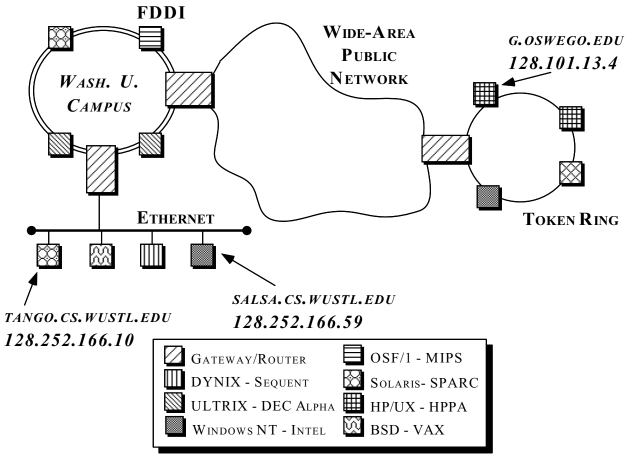
1. *Connectivity* (1970s)
   - Joining together end-systems into *networks*
   - Proprietary protocols
2. *Internetworking* (1980s)
   - Joining together networks into *internetworks*
   - Internetworking standards
     - ▷ Both *de facto* and *de jure*
3. *Interworking* (1990s)
   - Designing distributed computing systems that coordinate distributed applications in a robust, secure, flexible, and efficient manner
     - ▷ *e.g.*, "teamware," CSCW, DCE, etc.

Distributed Computing Systems

- DCSs contain policies and mechanisms for exchanging various classes of multimedia information across *heterogeneous* internetworks of gateways, bridges, and hosts
- Primary DCS Components

1. *Transmission media and network infrastructure*
2. *Communication protocols*
3. *Communication Models*
4. *Transport systems*
5. *Distributed application support*

Distributed Computing Systems (cont'd)



Symptoms of a Distributed Computing System

1. *Multiple Independent Processing Elements*
   - Each processing element possesses one or more CPUs and private memory
2. *Virtually Interconnected Hardware*
   - Typically "loosely-coupled"
   - Support network interprocess communication (IPC)
3. *Processing Elements Fail Independently*
   - Enhances fault tolerance
4. *Shared State*
   - Facilitates failure recovery
   - May be difficult/expensive to ensure

Challenges of DCSs

- *Technical*
  - Difficult and expensive to maintain shared or global information
  - Higher latency requires different algorithms and designs
    - ▷ *e.g.*, caching and process allocation
  - Debugging and performance profiling/tuning is challenging...
- *Administrative*
  - Authorization, authentication, and security issues are more problematic
- *Organizational*
  - Developers lack expertise with unfamiliar design and programming models

Motivations for DCS

- *Enhance computation capability via sharing of system resources, e.g.,*
  - File servers
  - Object managers and data bases
  - Processor pools
  - Various I/O devices, *e.g.*, printers, tape drives, modems, internetwork access points, etc.
- *Higher Fault Tolerance and Availability:*
  - Potentially higher availability and reliability, due to component redundancy and independent failure modes
  - Difficult to achieve in a fine-grain manner...

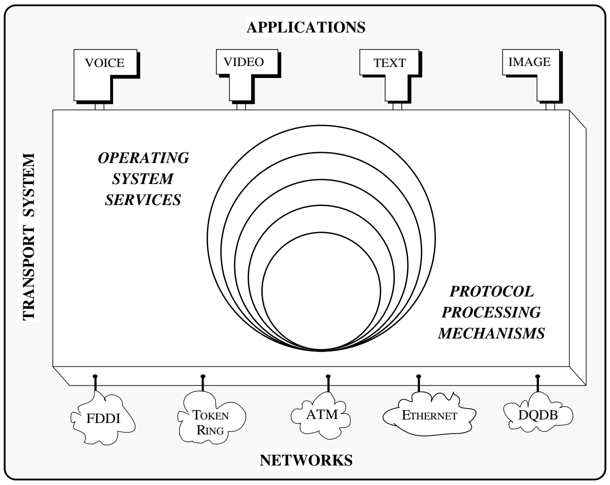
Motivations for DCS (cont'd)

- *Increase system capacity and reduce average response time:*
  - Potentially higher throughput
    - ▹ *e.g.*, utilizing coarse-grained parallelism
  - Increased scalability
    - ▹ Not limited by bus bandwidth
  - Oriented towards loosely-coupled multiprocessing applications
    - ▹ *e.g.*, parallel compilation and simulation
- *Better Price/Performance Ratios:*
  - Take advantage of price/performance improvements for systems hardware and communications technology
  - Migrate applications to most suitable resources

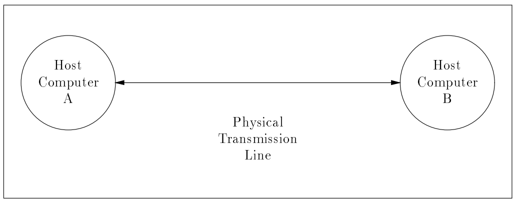
DCS in Practice

- Modern OSs (*e.g.*, UNIX, Windows NT) provide DCS environments that support distributed application development
  - Many earlier generation OSs did not...
- Many vendors are now pushing for standard support for distributed computing
  - *e.g.*, OMG CORBA, OSF's DCE, Sun's ONC, OLE/COM
- Progress has been relatively slow...

An Example DCS End System



Network Infrastructure



- A simple point-to-point network configuration
  - Note, hosts A and B may be arbitrary computers or other terminal devices
    - ▹ *e.g.*, frame buffers, video cameras, etc
  - Hosts may operate at different rates
  - Transmission line may be copper, fiber, microwave, radiowave, satellite, etc.

Network Infrastructure Challenges

- Even with this simple architecture, several types of problems may occur, involving:

  1. *Reliability*
  2. *Transmission Control*

- Moreover, as network topologies become more complex it is also necessary to handle:

  1. *Routing*
  2. *Congestion*

Reliability

- Transmission channels are not always error-free

- Therefore, depending on applications, we *may* need mechanisms

  * *Error detection*
  * *Error reporting*
  * *Error recover and correction*

- Common reliability management schemes for both bit- and packet-level errors include

  1. "Positive Acknowledgment with Retransmission" (PAR)
  2. "Automatic Repeat Request" (ARQ)
  3. "Forward error correction" (*e.g.*, Hamming code)
  4. "Ostrich Approach" (*i.e.*, do nothing)
     – Note "end-to-end" argument...

Transmission Control

- Two related problems

  1. Host A may send data at a faster rate than host B can handle it
  2. Multiple sender hosts may swamp resources at a single receiver host

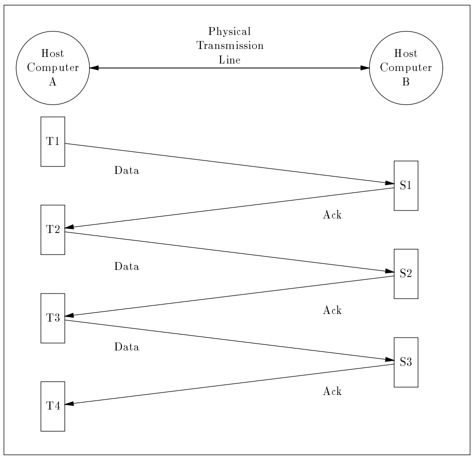
- Buffer overflows and CPU saturation result if either problem remains unchecked

Transmission Control (cont'd)

- Common solutions involve:

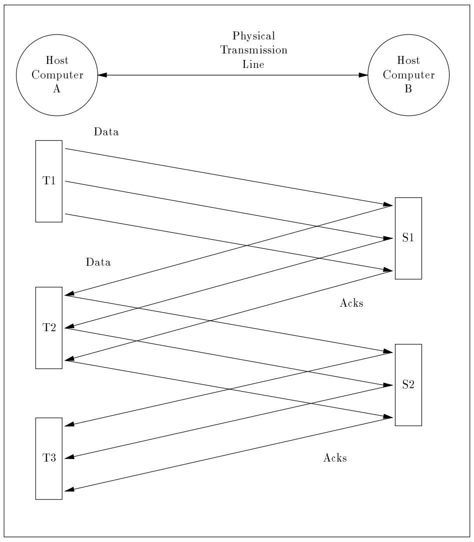
  1. *Flow Control* (reactive)
     (a) "Stop-and-wait" (ping-pong)
     (b) "Sliding window" (pipeline)
  2. *Rate Control* (preventive)
     – Controls the *burst amount* and *burst interval*

Stop-and-wait protocol



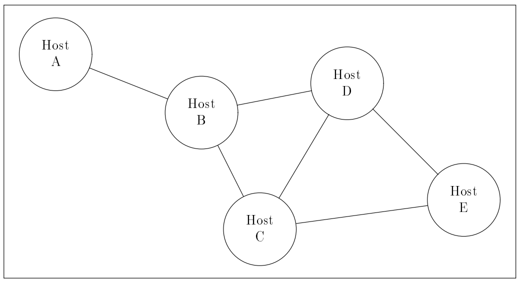
- Problem: wasted resources: CPU and network bandwidth are underutilized

Sliding window protocol



- Goal: fully utilize network and transport system during steady state
  - Challenge: how big should the window be?

Routing



- Complex network topologies require support for *routing*
  - If host A sends packets to host E then hosts B, C, and D have to make routing decisions
- Distinguish between routing:
  - *Policies* – *e.g.*, updating routing tables
  - 'Mechanisms – *e.g.*, how a route is located

Routing Schemes

- There are many types of routing schemes:
  1. *Fixed Routing*
     - Fast, but inflexible
  2. *Dynamic Routing*
     - Flexible, but less efficient
  3. *Hybrid Schemes*
- Note, routing behavior is usually transparent to distributed applications
  - *i.e.*, they do not have much, if any, control over routing behavior

Fixed Routing

- For a given source/destination pair, a path is given when configuring the network
- *Advantages*
  - Makes it easy to implement routers/gateways
  - May be efficient for certain situations, since route lookup algorithms may be optimized...
- *Disadvantages*
  - Not very robust or responsive when failures or congestion occurs

Dynamic Routing

- Gateways determine which path is most efficient at run-time
- *Advantages*
  - More adaptive to dynamic changes in network traffic and available routes
- *Disadvantages*
  - Per-packet routing overhead is very high
    - ▷ *i.e.*, more than just a table lookup is involved

Hybrid Routing

- Computers periodically exchange information and determine the "best" path to destination
  - Note, this is a good example of a distributed algorithm
- *Advantages*
  - Combines elements of both fixed and dynamic routing
  - Lookups are relatively fast
- *Disadvantages*
  - Periodic exchange of information creates extra traffic
  - Requires time to propagate information to other computers and gateways
  - "Routing loops" may occur

Congestion

- Congestion is a phenomena that occurs if host computers send more packets through a network than the intermediate gateway(s) are capable of handling
  - *e.g.*, the freeway during rush hour!
- Alleviating this problem requires congestion control algorithms to perform "traffic smoothing"
  - *e.g.*, *leaky-bucket* method, which controls how fast new packets are accepted into the network
    - ▷ *i.e.*, just like "access-control" traffic lights on freeway!

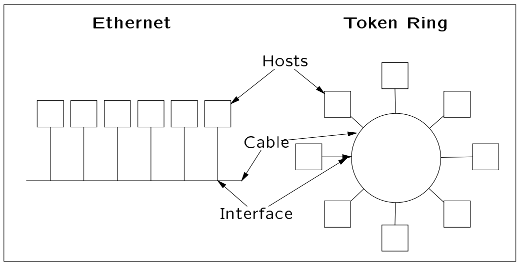
Congestion Control vs. Flow Control

- Congestion control is a *global* operation that protects shared buffers in the gateways
- Flow control is a *localized* operation that protects buffers in end host systems

Network Diameters

1. Local Area Networks (LAN)
   - Small distance between hosts
     - *e.g.*, within a building
2. Metropolitan Area Networks (MAN)
   - Somewhat large diameter
     - *e.g.*, typically within one organization
3. Wide Area Networks (WAN) remote networks
   - Large physical distances between hosts

Local Area Networks (LANs)



- Typical configuration of Local Area Networks
  - Note, the communication line is *shared* among multiple users
  - Collisions occur if more than one user sends a packet at the same time

Local Area Networks (LANs)

- *Ethernet* – (CSMA/CD)
  - 10 Mbps
  - Random access protocol
    - ▹ Send when you have a packet to send
    - ▹ If there is a collision, backoff and retransmit
    - ▹ “polite people in a dark room”
- *Token Ring*
  - 4/16 Mbps
  - Demand assignment
    - ▹ Assign channel capacity only to those who have packets to send

Metropolitan Area Networks (MANs) Example

- *FDDI* – (Fiber Distributed Data Interface)
  - 100 Mbps
  - Fiber optics, "dual counter-rotating ring topology"
  - Supports synchronous and asynchronous traffic
- *DQDB* – (Distributed Queue Dual Bus)
  - 160 Mbps
  - Bus topology
  - Supports isochronous and asynchronous traffic
  - Uses CCITT ATM packet format

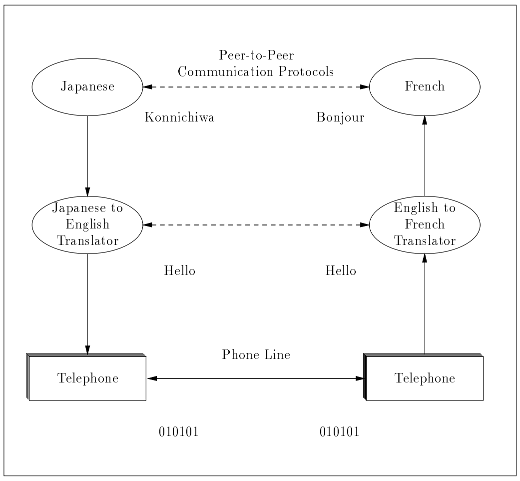
Wide Area Networks (WANs) Example

- *ATM* – (Asynchronous Transfer Mode)
  - 155/622 Mbps
  - Very high-speed packet switched, connection-oriented network
  - Based on optical transmission and VLSI technology
  - Multi-service support for multimedia applications
    - ▷ *e.g.*, voice, video, data, image
- *X.25*
  - Traditional data communications network architecture
  - Point-to-point

Communication Protocols

- A protocol is a set of rules or conventions governing how two or more entities cooperate to exchange data
- As shown previously, there is tremendous diversity of components in distributed computing systems
  - Therefore, we need standard protocols to communicate reliably, efficiently, and correctly
- If protocols are standard, then interoperability may be achieved even if all other aspects of network/computer communication are heterogeneous

Communication Protocols (cont'd)



Services vs. Protocols

- Services are operations *provided* to *consumers*
- Protocols implement these services
  - Protocols are accessed via "service interfaces"
- Distinguishing services from protocols enables providers to incorporate new technologies while maintaining backward compatibility at the "service interface" level

Services vs. Protocols (cont'd)

- Note, there may be:
  1. Multiple protocols for a given service
     - *e.g.*, "reliable stream communication"
  2. Multiple service interfaces for a given protocol
     - *e.g.*, BSD Sockets vs. System V TLI

Types of Service

1. *Non-reliable real-time*
   - *e.g.*, voice and video
2. *Reliable real-time*
   - *e.g.*, manufacturing control and robotics
3. *Non-reliable non-real-time*
   - *e.g.*, junk email
4. *Reliable non-real-time*
   - *e.g.*, bulk file transfer, remote login

Higher-layer Protocols

1. *Transport Protocols*
   - *e.g.*, TCP, XTP, TP4, UDP, RPC, VMTP
   - Basic categories include connection-oriented, connectionless, request/response
2. Distributed Object Protocols
   - *e.g.*, CORBA/DCE wire protocols

Higher-layer Protocols (cont'd)

1. *Distributed file system protocols*
   - NFS, AFS/DFS, RFS

2. *X-window protocols*
   - Used to decouple clients and *X* servers
   - Make window applications independent of the hardware; change hardware without changing applications

Protocol Specifications

- Protocols are specified by describing the objects and operations that comprise a particular communication protocol

- Protocol specifications describe:
  - *Protocol services and assumptions*
    - ▷ *e.g.*, file transfer, remote login, ARP over broadcast network
  - *Protocol vocabulary and encodings*
    - ▷ *e.g.*, packet types, header formats, packet sizes, byte ordering
  - *Procedure rules*
    - ▷ *e.g.*, state machine transitions
    - ▷ Schemes for connection, flow and congestion, and reliability management

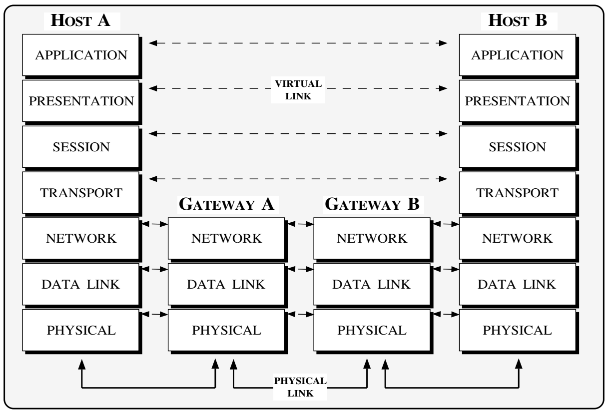
Communication Models

- To reduce complexity and enable vendor independence, communication protocols are commonly layered into a hierarchy that forms a "communication model"
  - *i.e.*, a *protocol stack* (or more generally, a *protocol graph*)

- A protocol graph represents the hierarchical relations between protocols in one or more "protocol suites"

- Well-defined protocol graphs exist for network protocol suites
  - *e.g.*, ISO OSI, TCP/IP, XNS, Novell, SNA

ISO OSI 7 layer Reference Model



Higher layer protocols in OSI

- (7) *Application Layer*
  - Standard remote services like file transfer, directory services, mail, etc
- (6) *Presentation Layer*
  - Encryption, basic encoding rules (*e.g.*, network/host byte-ordering), compression
  - XDR and ASN.1
- (5) *Session Layer*
  - "Dialog management"
- (4) *Transport Layer*
  - Services that ensure end-to-end communication
    - ▷ *e.g.*, reliable, in-order, non-duplicated data delivery

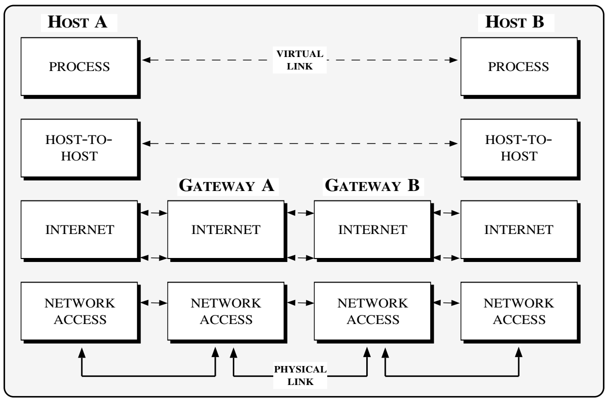
Lower layer protocols in OSI Reference Model

- (3) *Network Layer*
  - Routing, congestion control, fragmentation and reassembly
- (2) *Data Link Layer*
  - Services for hop-to-hop communication
    - ▷ *e.g.*, frame creation, frame error control
- (1) *Physical Layer*
  - Hardware (*e.g.*, # of pins, voltage, etc.)

Throughput Preservation Problem

- Note, network performance has improved by 5 to 6 orders of magnitude (from kbps to Gbps)
- Due to advances in fiber optics and VLSI technology
- Note, mostly hardware at this level...

Internet Communication Model

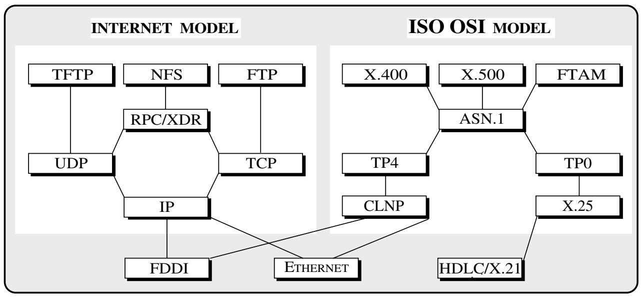


- Note, applications are responsible for *application*, *presentation*, and *session* layer services
  - Increases software redundancy, but may improve performance

OSI vs. Internet

1. *OSI*
   - The *de jure* standard
   - International in its scope
   - *Top-down* specification/development process
     - "Committees do research, researchers do implementation"
   - Intended as "displacement technology"
     - *i.e.*, "installed-base hostile"
2. *Internet*
   - The *de facto* standard
   - Not vendor-specific
   - Research, implement, deploy, and test before standardization!
   - Motto: *respect the installed base*

Comparision of Internet and OSI Protocol Families



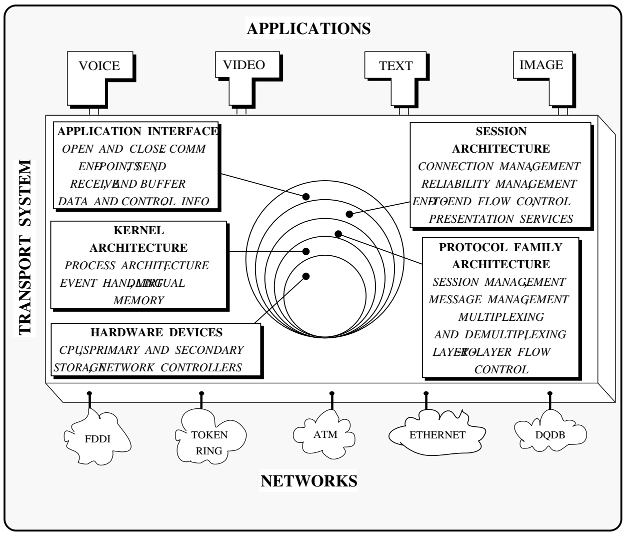
Communication Subsystems

- Communication protocols do not generally exist in a vacuum
  - Instead, they exist within an integration framework offered by the *communication subsystem*
- A communication subsystem combines
  1. *Communication protocol tasks*
     - Such as connection management, data transmission control, remote context management, and error protection
  2. *Operating system services*
     - Such as memory and process management
  3. *Network hardware components*
     - Local, metropolitan, and wide area networks devices

Communication Subsystem (cont'd)

- Communication subsystems integrate network protocols (*e.g.*, TCP, TP4, VMTP, XTP) into the operating systems of host computers
  - Typically incorporates OSI layers 3 and above
- Computer architecture performance has improved overall by 2 to 3 orders of magnitude (from 1 MIP to 100 MIPS)
  - Due to hardware advances such as (1) faster clock speeds, (2) larger caches, (3) *superpipelining*, and (4) *superscalar* architectures
- However, communication subsystems are largely written in software
  - Therefore, their overall improvement has not been 2 to 3 orders of magnitude, leading to a *throughput preservation problem*

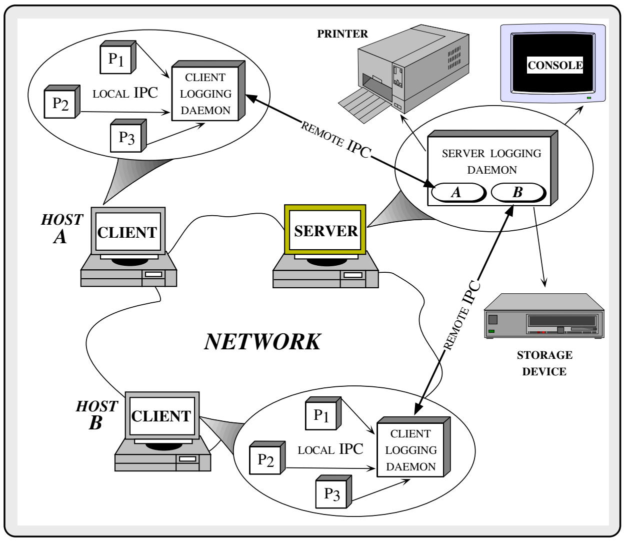
Communication Subsystem (cont'd)



Distributed Applications

- Many applications and systems benefit from using distributed programming techniques
  - *e.g.*,, data bases, on-line transaction processing, network file systems, real-time process monitoring systems
- Reasons include:
  - Share expensive resources
  - Enhance fault tolerance
  - Provide opportunity to exploit available parallelism
  - Increased scalability
  - Improve modularity by reducing data coupling
  - Reduce cost

Distributed Applications (cont'd)



Distributed Applications (cont'd)

- Distributed applications are often more complicated and difficult to develop due to
  - Diversity of applications, communication subsystems, host/network interfaces, networks
  - Network complexity is often greater than the sum of its parts (each of which may be relatively simple), *e.g.*,
    - ▷ Distributed state and continual change
    - ▷ Debugging is more complicated
    - ▷ Subtle timing issues
  - Trade-offs between modularity and efficiency
  - Client/Server Asymmetries
    - ▷ *e.g.*, difference between asynchronous servers and synchronous clients
      - · Multiplexing/demultiplexing
      - · Concurrency

Distributed Applications (cont'd)

- Many integrated tool suites and reusable libraries are being developed to facilitate the development of distributed applications
  - *e.g.*, ONC RPC and OSF DCE

- Services include
  - *Authentication, authorization, and data security*
  - *Remote service binding* (*i.e.*, naming and identification)
  - *Service registration*
  - *Presentation layer conversion*
  - *Remote communication*
  - *Automatic dispatching of pre-registered services*
    - ▹ *i.e.*, separates "policy and mechanisms" for typical event-driven applications

Distributed Applications (cont'd)

- An important goal of the tools, libraries, and environments is often to make distribution transparent...
  - *e.g.*, Remote Procedure Calls RPC

- In addition, these infrastructures also provide many reusable mechanisms and abstractions that simplify distributed application development

Summary

- Distributed computing systems are becoming increasingly essential in research and commercial settings

- It is important to understand the networking aspects, communication subsystem aspects, and application aspects of these distributed systems

- Communication protocols are used to control diversity
  - Standardized communication protocols aid in the development of portable distributed applications by allowing for independence from specific
    1. *Hardware platforms*
    2. *Vendors*
  - However, there are several standards...