CompSci 131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- **Last lecture review**
- **Types of Distributed Systems**

- **Reading assignment:**
  - **Today's: Sec. 1.3.1-1.3.2**
  - **Next: 2.1-2.2**
    - » **Complete the assignment *before* next class**

Summary of last lecture

- **A distributed system is a collection of independent computers that appears to its users as a single coherent system.**

- **Design goals**
  - **Allow users to easily connect to resources**
  - **Hide the fact that resources are distributed (Transparency)**
  - **Openness to allow extendibility, interoperability**
  - **Scalability to allow growth without performance loss**

‘First Time’ DS Design Pitfalls

- Are standard sftw engineering rules enough?
- No, pitfalls include assuming that
  - Network is reliable
  - Network is secure
  - Network is homogeneous
  - Topology does not change
  - Latency is zero
  - Bandwidth is infinite
  - Transport cost is zero
  - There is only one administrator
- Most of the DS design principles relate to the above
  - Solve problems caused by one or more of these being false

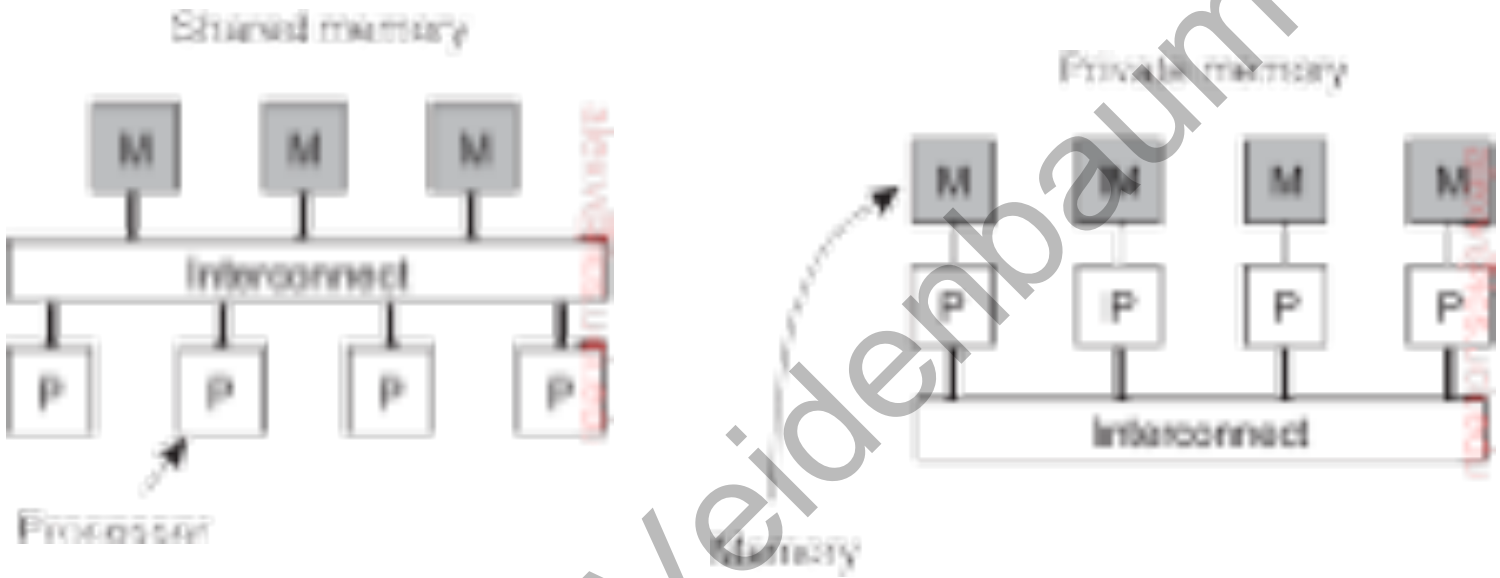
Types of Distributed Systems

- **Distributed Computing Systems**
  - Targets increase in application performance
- **Distributed Information Systems**
  - Supports enterprise computing
- **Distributed Pervasive Systems**
  - Collections of small devices, not even always on
  - Adaptable to local environment

Distributed Computing

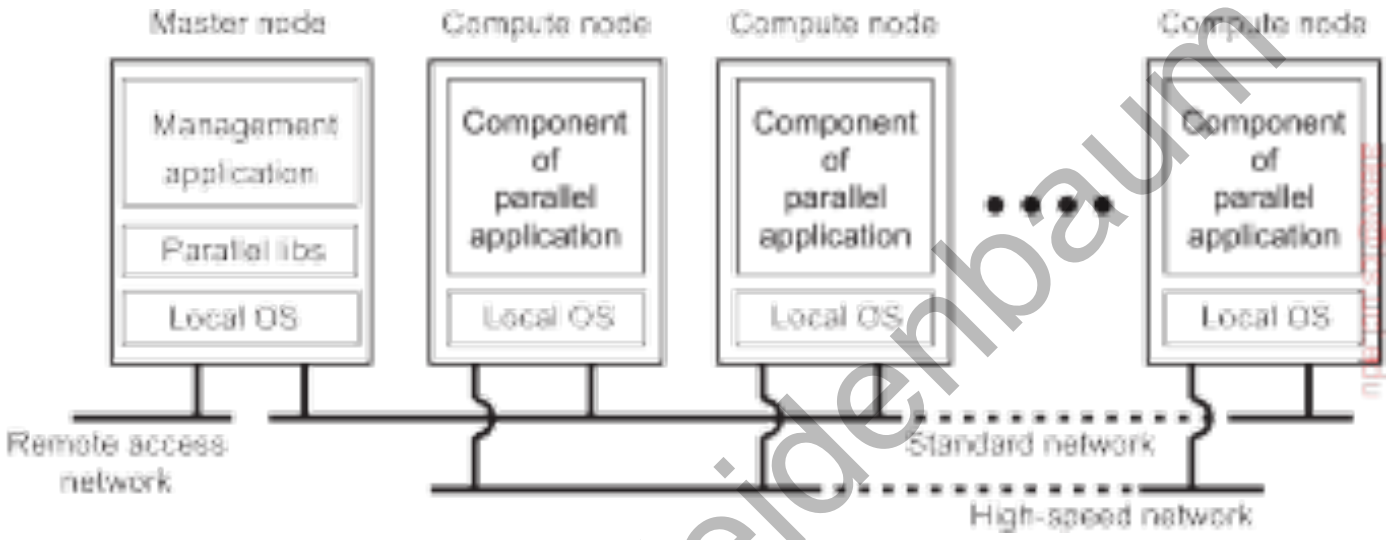
- **Clusters - integrate a small number of local systems**
  - **Master node: parallel libraries, application management**
  - **Multiple computing nodes**
  - **Middleware layer to tie together**
- **Small can mean 128 or even a few K nodes**
  - **ICS openlab is an example**
- **You will write a program using a middleware layer called MPI**
  - **Submit via a master queue using SLURP sftw**
- **Can be considered a small version a supercomputer**

Multiprocessor vs multi-computer



- Today high-performance computing nodes are multiprocessors
- Clusters are multi-computers
- Supercomputers are very large clusters with custom interconnect

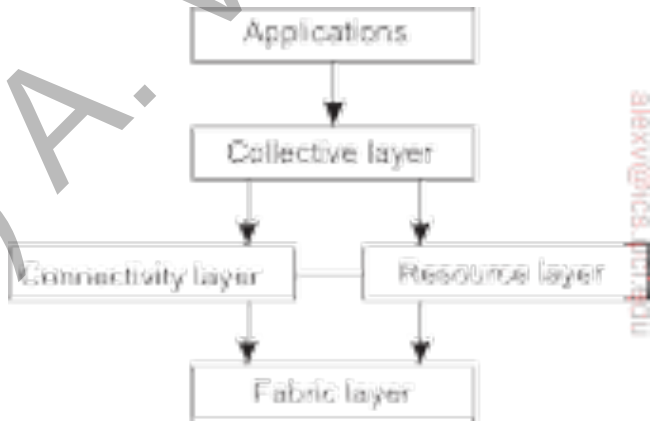
Cluster Computing



- **Only master is remotely accessible**
  - **Job submission, file exchange, etc**
- **Compute nodes run independent portions of a parallel aplication**

Grid computing

- **Integrate resources on the internet**
  - **Another layered software organization**
    - » **Fabric layer: interface to local resources at a given site**
    - » **Communication layer**
      - **Authentication, communication**
    - » **Resource layer: allows access to a given resource**
      - **Access control, configuration data access, specific operations**
    - » **Collective layer: access to multiple resources**
      - **Resource discovery, allocation, scheduling**
    - » **Application layer**



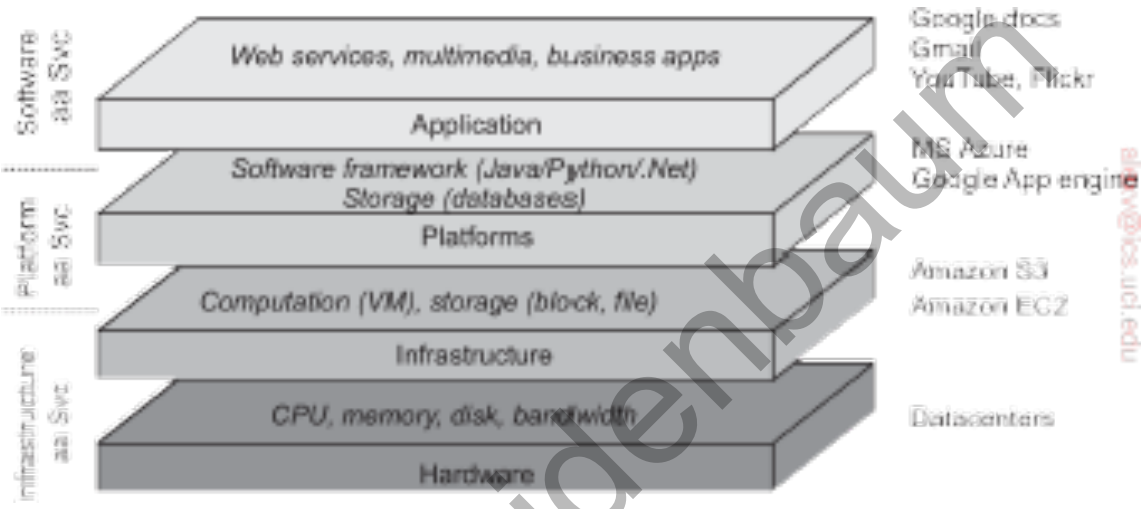
Grid computing

- **Integrates resources from different organization**
  - **Another name – federation**
  - **Creates a virtual organization for collaboration**
- ***Pre-cloud,* allowed many more nodes than a cluster**
- **What is the main difference with cluster?**
  - **other than size**

Cloud computing

- **Grid -> utility computing -> clouds**
  - **Another name – federation**
  - **Creates a virtual organization for collaboration**
- ***A cloud is a pool of virtualized resources***
- **Typically four layers**
  - **Hardware**
  - **Infrastructure (employs vitualization)**
  - **Platform (somewhat similar to OS: job submission, files)**
  - **Application (Google docs, Acrobat DC, etc)**

Cloud computing



- **These layers are offered to customers as**
  - **Infrastructure as a SVC (IaaS)**
  - **Platform as a SVC (PaaS)**
  - **Software as a SVC (Saas)**

Distributed Information Systems

- **Typically enterprise level systems**
  - **Often separating processing from data**
- **Transaction processing systems deal with data**

Transaction Processing Systems

- **Transactions protect operations on *data***
  - **Access/modify many data items in one atomic operation**
  - **An All or Nothing operation**
    - » **Either all data is accessed/modified or everything is aborted**
      - **Aborted = no state change occurs**
- **The origin is in business transactions**
  - **Negotiation followed by contract**
- **Distributed system' transactions are similar**
  - **Negotiate, perform, commit**
  - **back out on crashes, other problems**

Programming Transactions

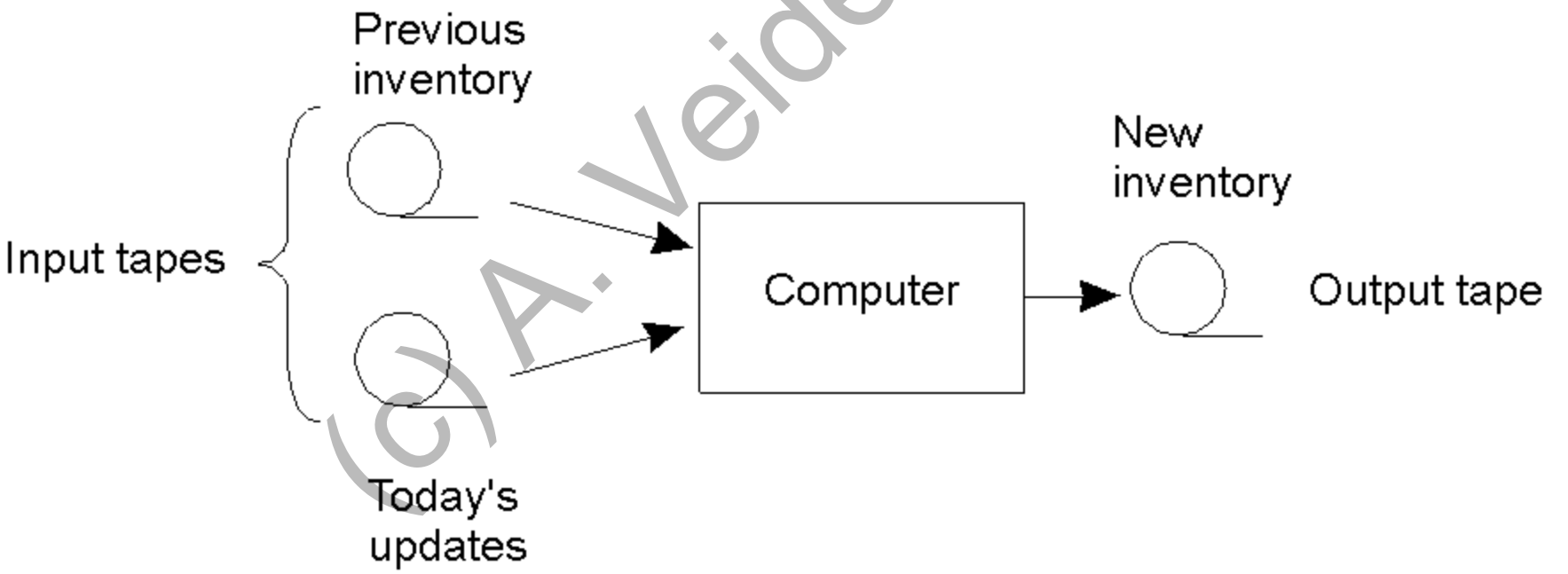
- **Basic operations to implement transactions**

Primitive	Description
BEGIN_TRANSACTION	**Make the start of a transaction**
END_TRANSACTION	**Terminate the transaction and try to commit**
ABORT_TRANSACTION	**Kill the transaction and restore the old values**
READ	**Read data from a file, a table, or otherwise**
WRITE	**Write data to a file, a table, or otherwise**

- **Transactions may be nested!**

Example from the Old Days

- **Store Inventory computation**
  - **Input: tapes of last inventory, today's sales+deliveries**
- **Computation can be aborted at any time**
  - **Rewind tapes, start over**



Today's example

- Online banking
  - Withdraw $1000 from account A
  - Deposit $1000 into account B
- What is the problem here?
- Can this be done using mutual exclusion?
- Transaction model
  - Perform both or abort

Properties of Transactions

1. **Atomic**
   - All or nothing, looks like everything happens at once
2. **Consistent**
   - Maintains system invariants
     - During a transaction may not be consistent
       - No one can observe that!
3. **Isolated or serializable**
   - Concurrent transactions do not affect each other
   - Result is as if they ran sequentially in some order
4. **Durable**
   - After commit the results ARE made permanent

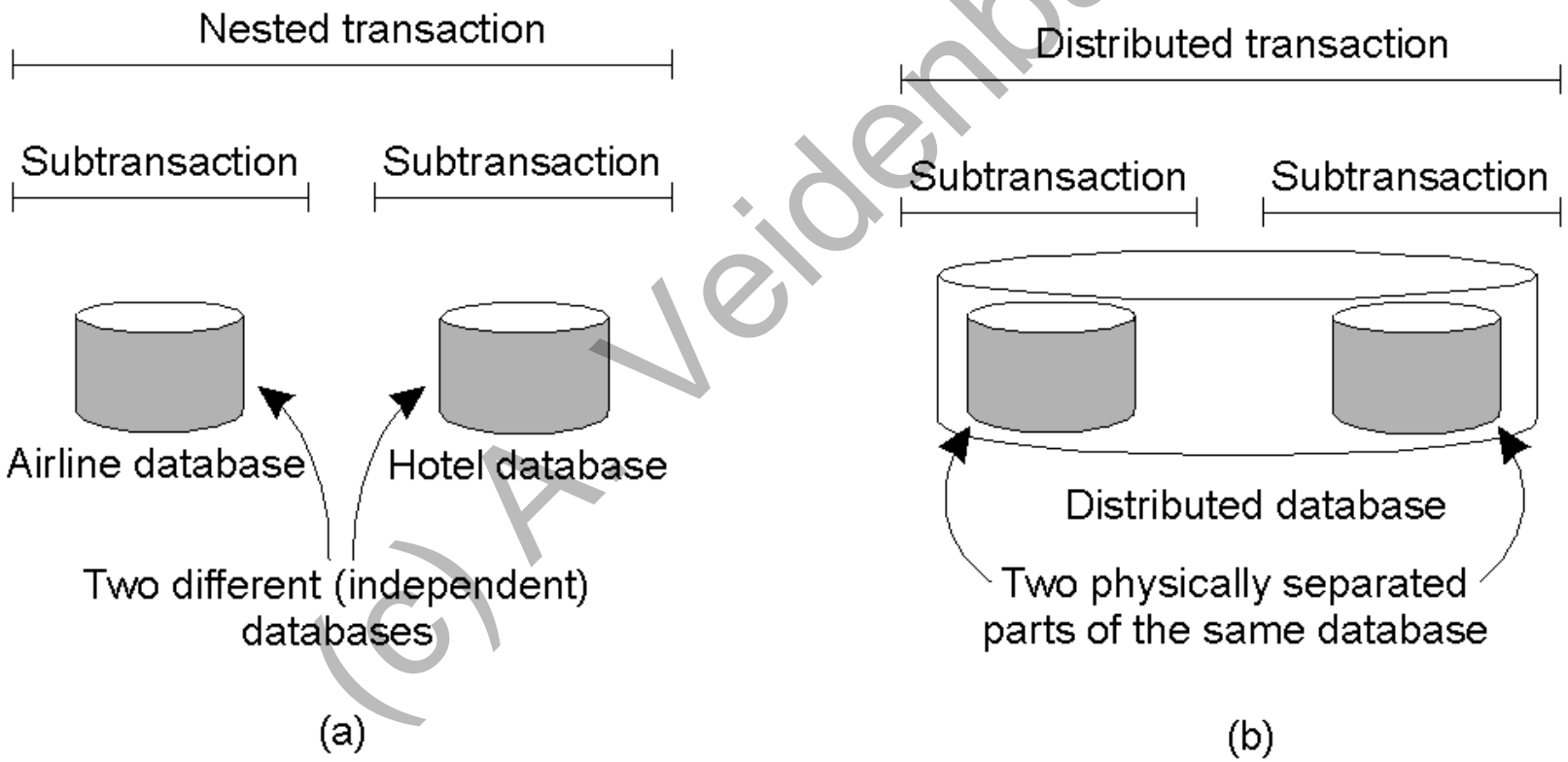
Transaction Models

- **Flat - a series of operations run to completion**
  - **Or aborted**
  - **Has limitations**
- **Nested**
  - **Top-level transaction can fork off child processes**
  - **Children can execute their own transactions**
    - » **Commit of a child transaction can be undone by parent**
- **Distributed**
  - **A flat transaction on distributed data**
  - **Requires distributed locking of data, commit**

Examples

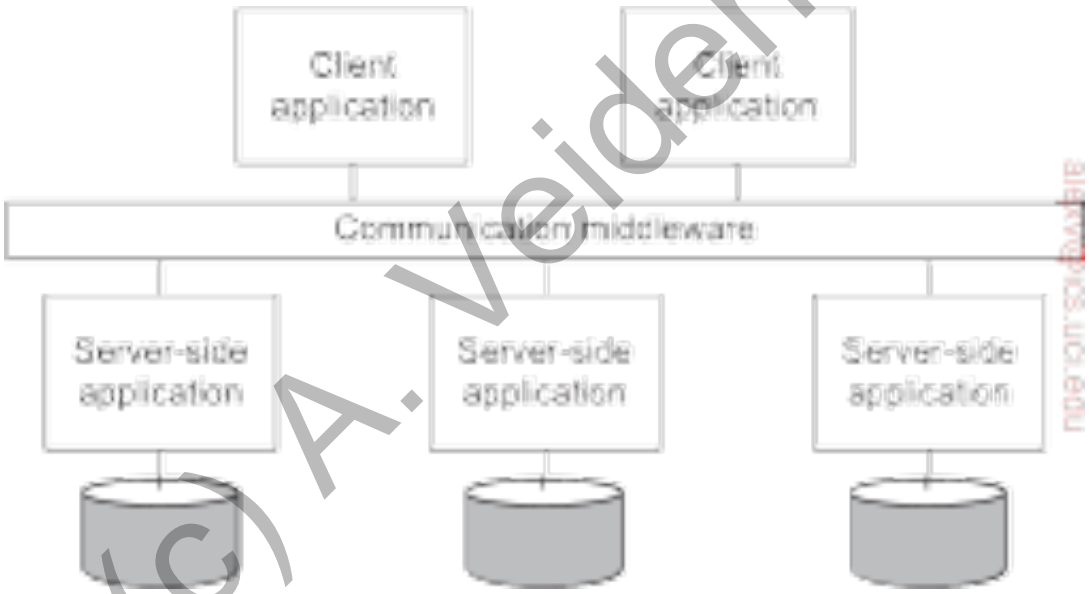
a) **Nested transactions - logically separate**

b) **Distributed transactions - on separate data**



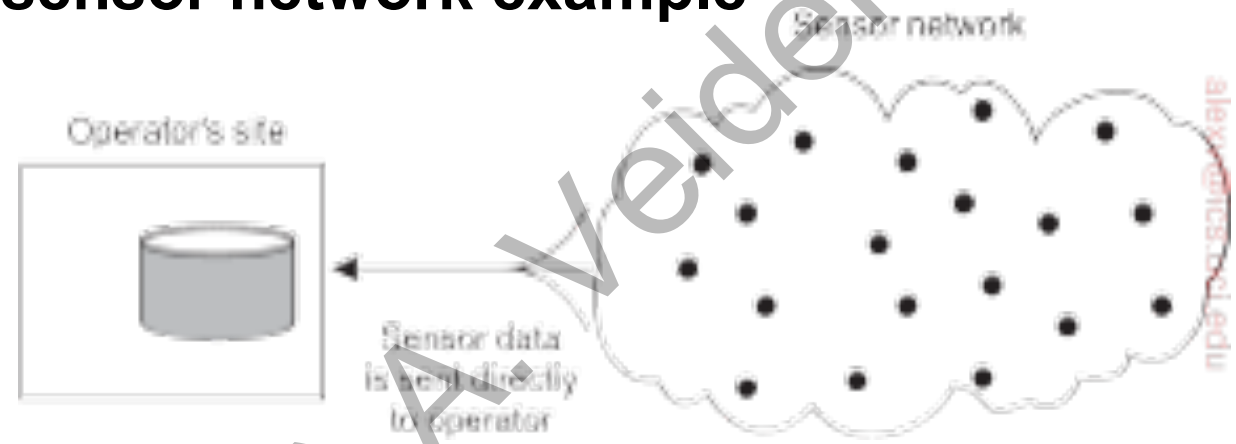
Enterprise application integration

- **Application components can communicate directly:**
  - **Need communication middleware**
    - » **Remote procedure call (RPC)**
    - » **Remote method invocation (RMI) for objects**



Pervasive Systems

- A pervasive system blends into our environment
  - » Mobile
  - » Sensor networks
- Nodes may move or not be up all the time
- A sensor network example



  - – All data processed by operator
    - » Sensors may form an ad–hoc network