Distributed Operating Systems

CIS-4020
Vermont Technical College
Peter C. Chapin

Distributed Characteristics

- Node Characteristics
  - Heterogeneous or homogeneous?
  - Autonomous or specialized?
  - One security domain or many domains?
- Communication
  - Conventional network or dedicated network?
  - Slow communication or fast communication?
  - General protocols or specialized protocols?
- Applications
  - Nodes work on same problem or many problems?

The Internet

- The largest distributed system on Earth.
  - Nodes heterogeneous
  - Nodes autonomous
  - Spans many security domains
  - Conventional network interconnections
  - Slow communication speed (in general)
  - General purpose network protocols
  - Works on many problems simultaneously

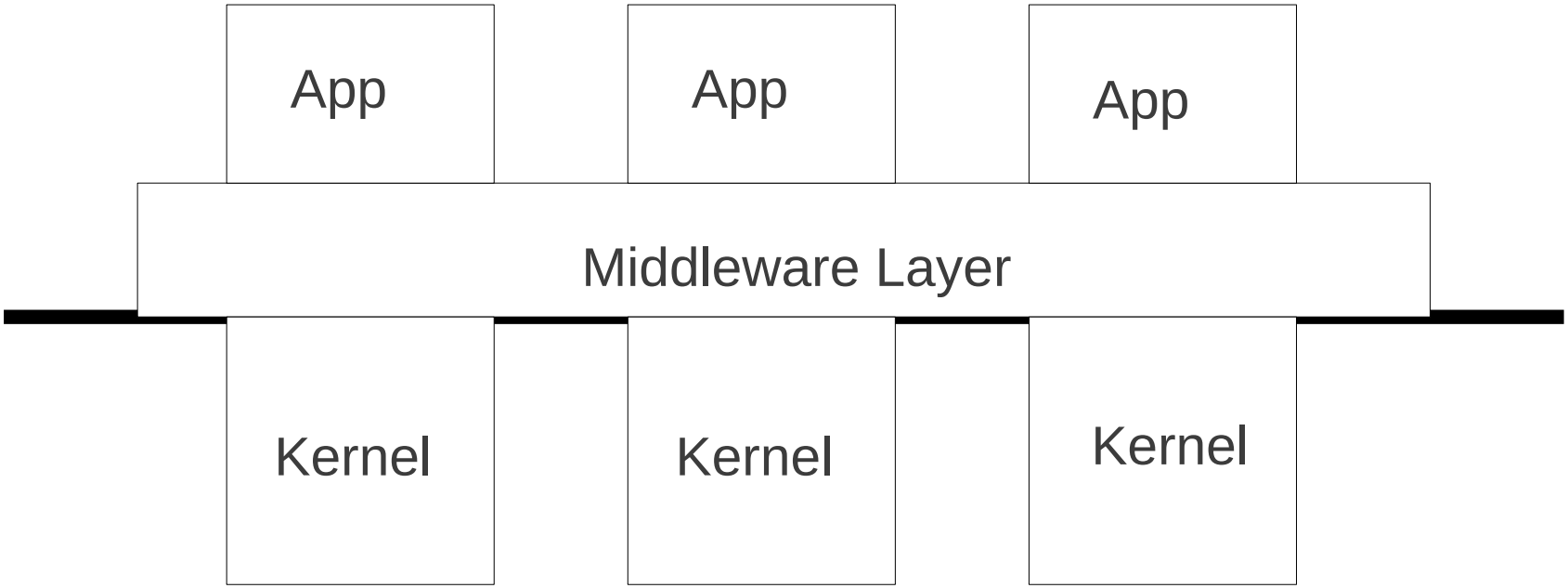
Clusters

- Supercomputers today are usually clusters of slower, less expensive machines.
  - Nodes homogeneous
  - Nodes autonomous/specialized (depending)
  - One security domain
  - Dedicated network
  - Fast communication
  - Specialized protocols
  - All nodes work on the same problem

Local Area Network

- A LAN (such as at VTC) is a kind of distributed system.
  - Nodes homogeneous (although not always)
  - Nodes autonomous (clients), specialized (servers)
  - One security domain
  - Conventional network
  - Slow communications
  - General purpose protocols
  - Nodes work on different problems

Distributed Applications

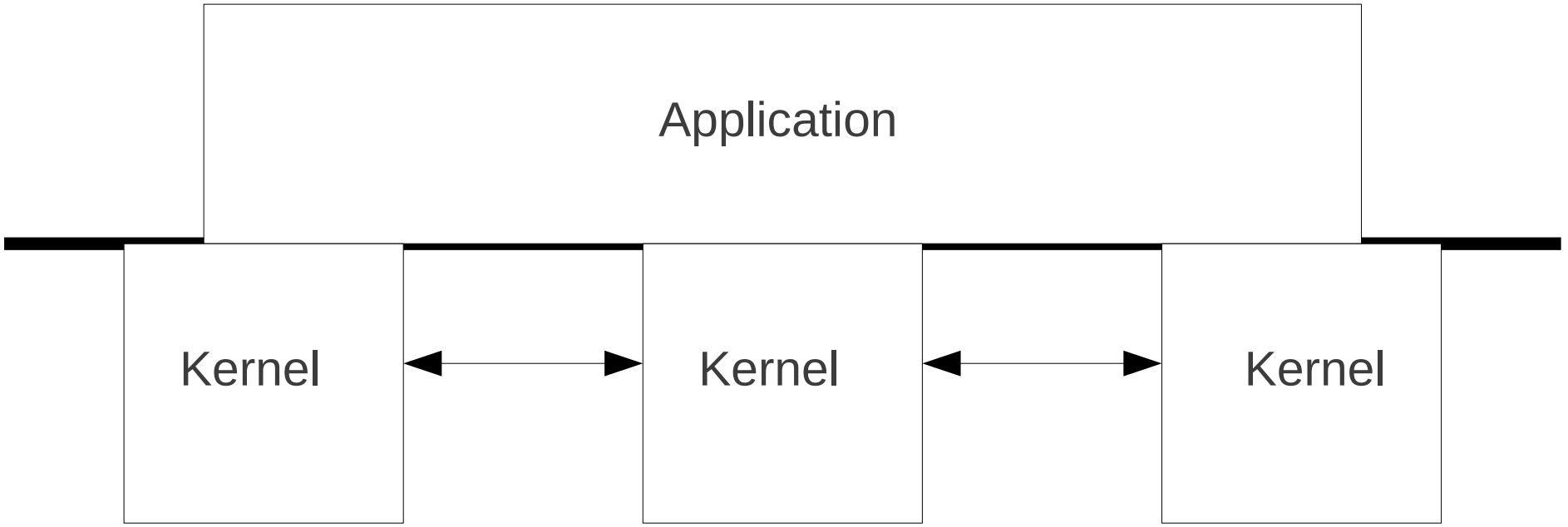


+ Operating systems are autonomous.
+ Separate systems integrated by middleware library.
+ Operating system does not “see” distributed system.

Distributed App Frameworks

- Java VM, .NET
  - Virtual machine based (homogeneous platform)
- CORBA, Ice, Web services
  - Library with standardized communication protocols
- Ada distributed systems annex
  - Programming language library (and language constructs)
- PVM, MPI
  - Library with server support. MPI is specialized for numerical work and common on supercomputers.

Distributed Operating System



+ Kernels communicate to integrate themselves into a single system.
+ Application does not “see” the distributed operating system.
+ Application sees illusion of a single machine.

Illusion of a Single Machine

- The distributed OS ideal
  - Single log on
  - Single file system
  - Single process ID space
  - Single I/O space (all I/O devices look local)
  - Single job management (migration?)
  - Single network presence
  - Single memory space (distributed shared memory)
- Some of these features are still research topics.

Distributed OS Examples

- Many systems have *some* distributes features.
- QNX
  - Allows common access to I/O devices
  - Has network transparent IPC
- Linux clusters
  - Beowulf http://www.beowulf.org/
- Plan9
  - Experimental OS from Bell Labs
  - http://plan9.bell-labs.com/plan9/

Security Domains

- Easier if there is only one security domain
  - Single authority for account information; makes access control easier.
  - Easier to enforce policy and configuration choices over the distributed system.
- Multiple domains are more interesting!
  - Some resources are owned by foreign domains.
  - Very large systems inevitably span security domains.

Node Availability

- Nodes always on.
  - Simplifies integration
    - Unified file system
    - I/O devices always available
    - Node's memory always available
  - Simplifies programming
    - Number of nodes fixed and known ahead of time.
- Nodes turn off and on.
  - Must migrate resources in use before shutdown.
  - Programs must adapt to the number of nodes.
  - Ideally the OS would take care of these things!

Network Availability

- Conventional networks.
  - Sometimes fail
    - Distributed system is partitioned.
    - Nodes must cope with being isolated now and then.
  - Variable performance
    - High traffic (from other sources) interferes with system.
- Dedicated networks.
  - More reliable.
  - More uniform performance.
  - Better potential performance.

Resource Availability

- Specialized resources
  - Unusual I/O devices
  - Unusual external hardware connected to a distinguished node (sensor, controller, etc)
  - Unusual computational resources (co-processors, video cards, etc).
- Systems based on homogeneous nodes
  - Can have problems coping with specialized resources.
  - Workaround: Master/slaves configuration
    - Master has all unique I/O devices. Slaves just compute.

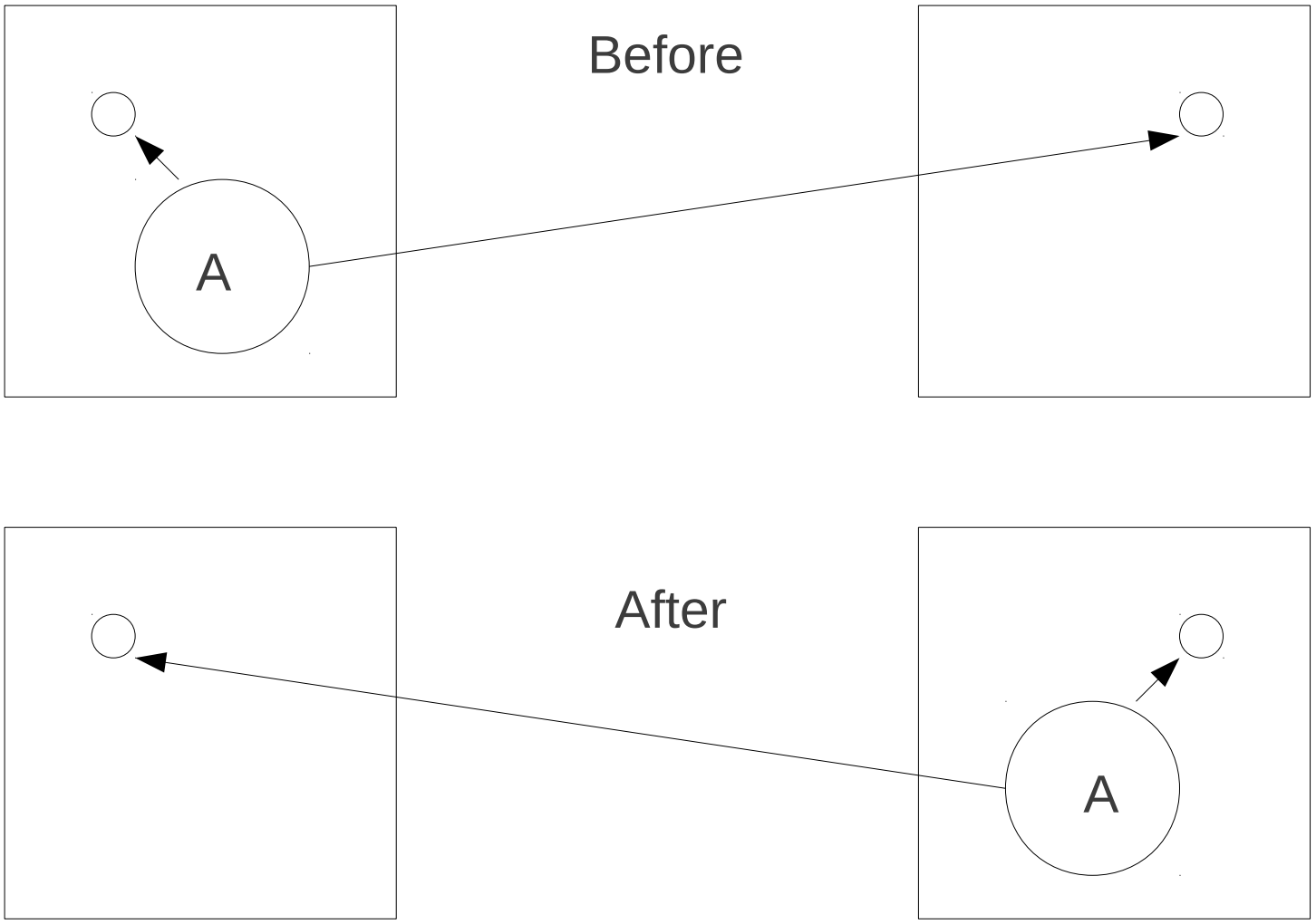
Process Migration

- *Defn*: Moving a process between nodes.
- Why do it?
  - Load balancing
  - Communication performance
    - Move the process to the data
  - Availability
    - Move a process before shutting down a system
  - Access to specialized resources
    - Move a process to the resource

Who Decides to Migrate?

- User
  - Issues commands to move process.
    - Assumes the user knows what he/she is doing.
- Application
  - Program calls API requesting to be moved.
    - Requires a way for the program to query configuration.
- Operating System
  - Decision is automatic
    - Most difficult to do well.

Migration and IPC



IPC must be network transparent

Migration is Hard

- Move entire address space?
  - Demand paging over the network?
- Move OS state information about process.
- Save to disk and then page from file over net?
- What about queued messages and signals?