Process

Distributed Systems IT332

Outline

- Threads
- Virtualization
- Server
- Code Migration

Processes and Threads

Process

- An instance of a computer program that is being executed on one of the operating system's (virtual) processors .
  - While a program itself is just a passive collection of instructions, a process is the actual execution of those instructions

- A process has a virtual address space, executable code, open handles to system objects, a security context, a unique identifier, environment variables, a priority class, and at least one thread of execution.

Processes and Threads

Threads

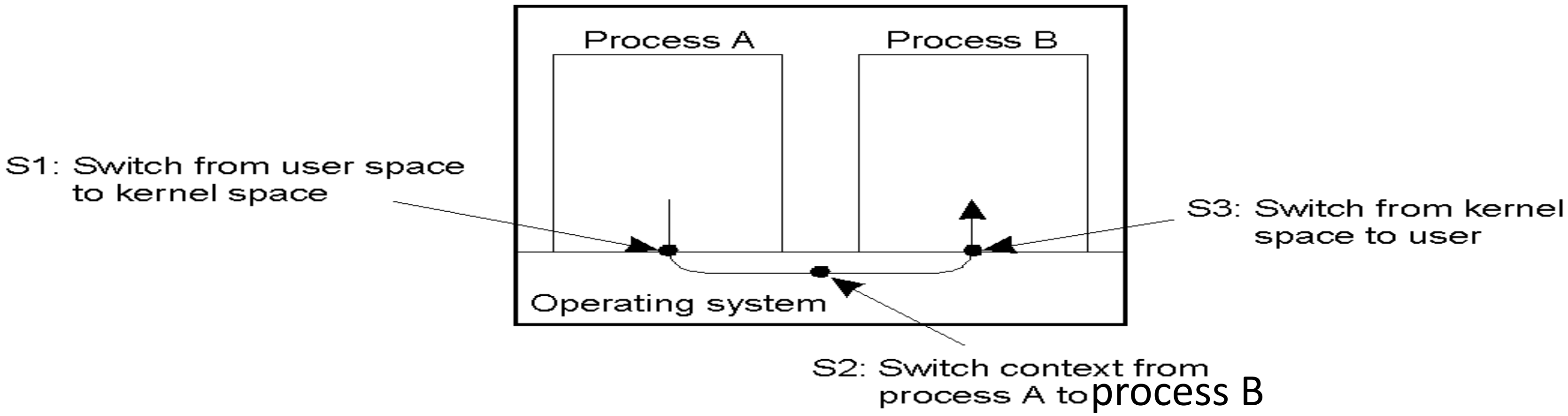
- Each process is started with a single thread, often called the primary thread, but can create additional threads from any of its threads.
- Thread (lightweight process)
  - A thread is a path of execution within a process. A thread executes its own piece of code, independently from other threads.
  - A way for a process to split itself into two or more simultaneously running tasks.
  - A thread maintains only the minimum information to allow a CPU to be shared by several threads. ⍰
  - In particular, a thread context often consists of nothing more than the CPU context.

Processes and Threads

Advantages of Threads

- Allow for parallel computation.
- Switching between threads is faster than switching between process.
- Creating and destroying threads is cheaper than creating and destroying a process since that:
  - Threads use very little resources of an operating system in which they are working.
  - Threads do not need new address space, global data, program code or operating system resources and can share common data
- Easier to structure many applications as a collection of cooperating threads
- Higher performance compared to multiple processes since switching between threads takes less time.

Thread Usage in Non-Distributed Systems


Process A
Process B
S1: Switch from user space to kernel space
S3: Switch from kernel space to user
Operating system
S2: Switch context from process A to process B

Context switching as the result of IPC.

- The major drawback of all IPC mechanisms is that communication often requires extensive context switching, shown at three different points.

Thread Usage in Non-Distributed Systems

- Example: A spreadsheet program where the different cells are dependent, when a user changes the value in a single cell, such a modification can trigger a large series of computations:
  - If there is only a single thread of control, computation cannot proceed while the program is waiting for input. Likewise, it is not easy to provide input while dependencies are being calculated.

- → The easy solution is to have at least two threads of control: one for handling interaction with the user and one for updating the spreadsheet. In the mean time, a third thread could be used for backing up the spreadsheet to disk while the other two are doing their work.

Threads in Distributed Systems

- Used to express communication in the form of multiple logical connections at the same time.
- An important property of threads is that they can provide a convenient means of allowing blocking system calls without blocking the entire process in which the thread is running.
- A main contribution of threads in distributed systems is that they allow clients and servers to be constructed such that communication and local processing can overlap, resulting in a high level of performance.
- Attractive to use in distributed systems
  - Multithreaded client
  - Multithreaded server

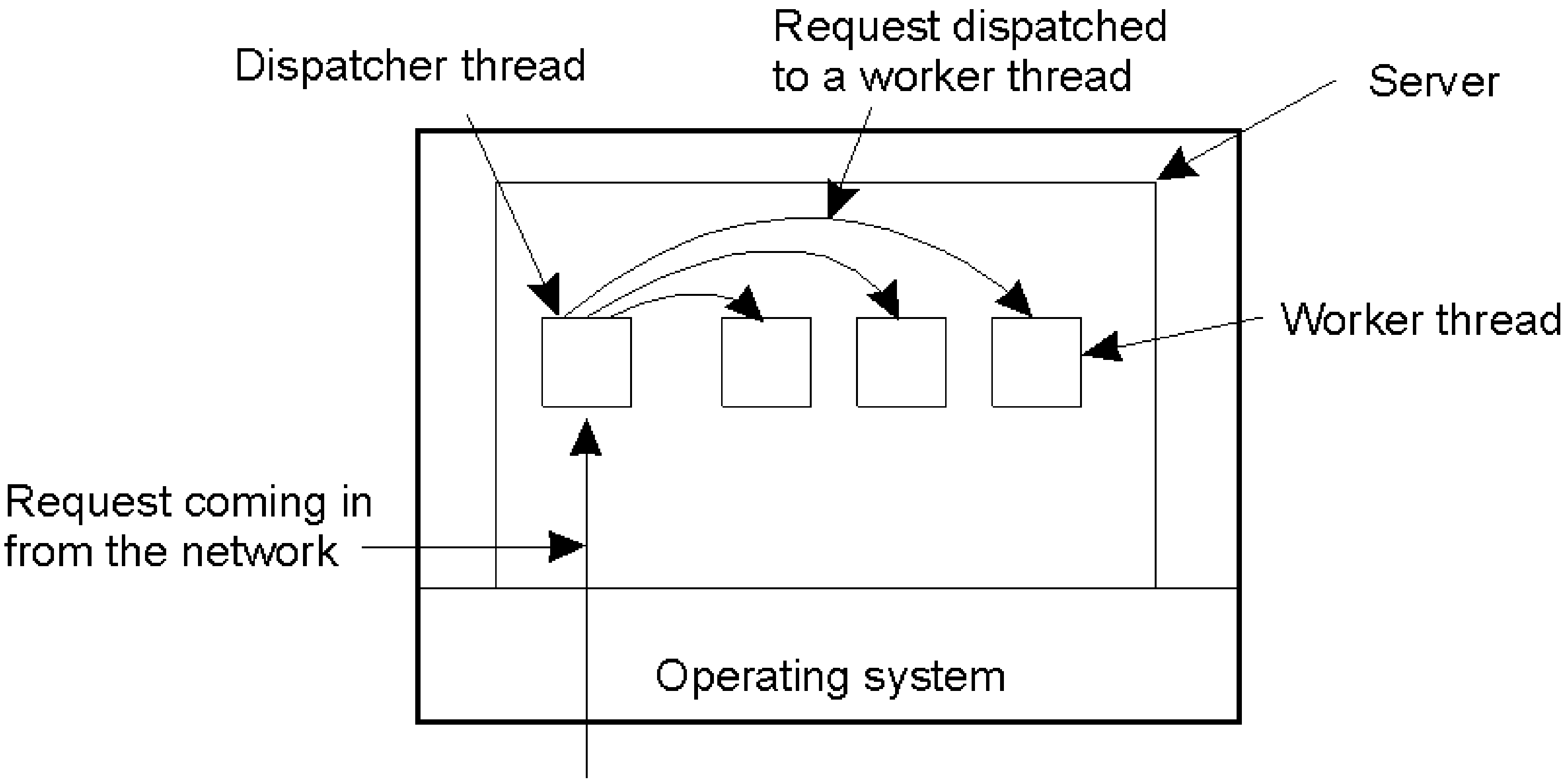
Multithreaded Clients

- Can be used to hide delays/latencies in network communications, by initiating communication and immediately proceeding with something else.

- Example: web browsers such as IE are multi-threaded
  - A web browser can start up several threads: once the main HTML file has been fetched, separate threads can be activated to take care of fetching the other parts. Each thread sets up a separate connection to the server and pulls in the data. One for downloading the HTML source of the page, one each for images on the page, one each for animations/applets etc.

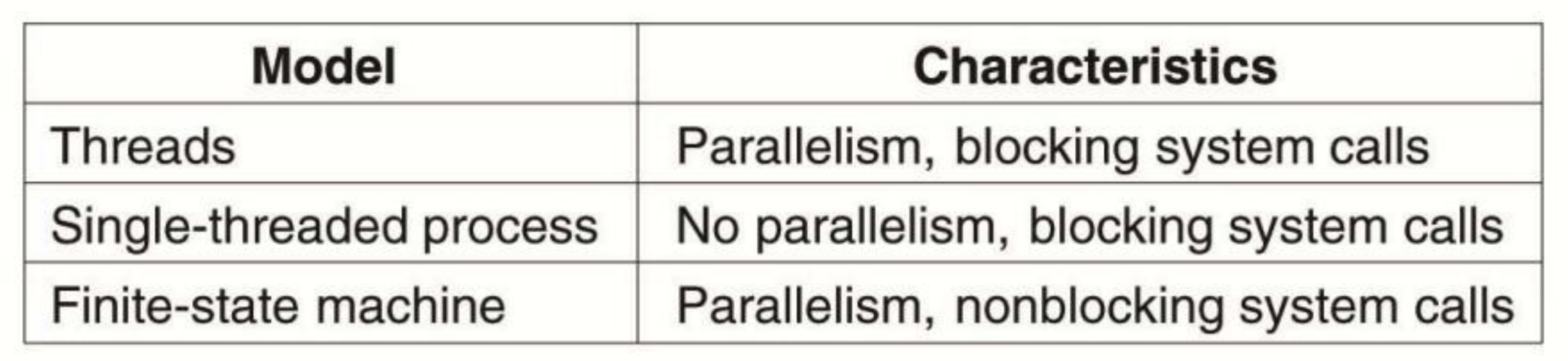
- Replicated web servers along with multi-threaded clients can result in shorter download times.

Multithreaded Servers



A multithreaded server organized in a dispatcher/worker model

Multithreaded Servers



Model	Characteristics
Threads	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls

Three ways to construct a server

- Blocking system calls make programming easier and parallelism improves performance.

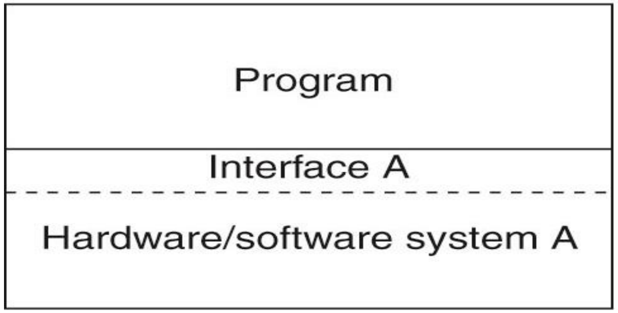
- The single-threaded server retains the ease and simplicity of blocking system calls, but gives up performance.

- The finite-state machine approach achieves high performance through parallelism, and uses nonblocking calls, thus is hard to program.

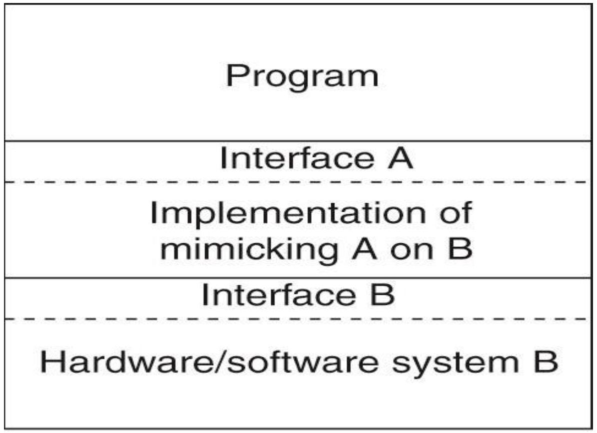
Virtualization

- Virtualization is the creation of a virtual (rather than actual) version of something, such as a hardware platform, resources, operating system, a storage device or network resources.

- Introduced in 1970s. IBM have applied this technique very successfully for a long time on the IBM 370 mainframes (and their successors): to offer a virtual machine to which different OS had been ported.



(a) General organization between a program, interface, and system.

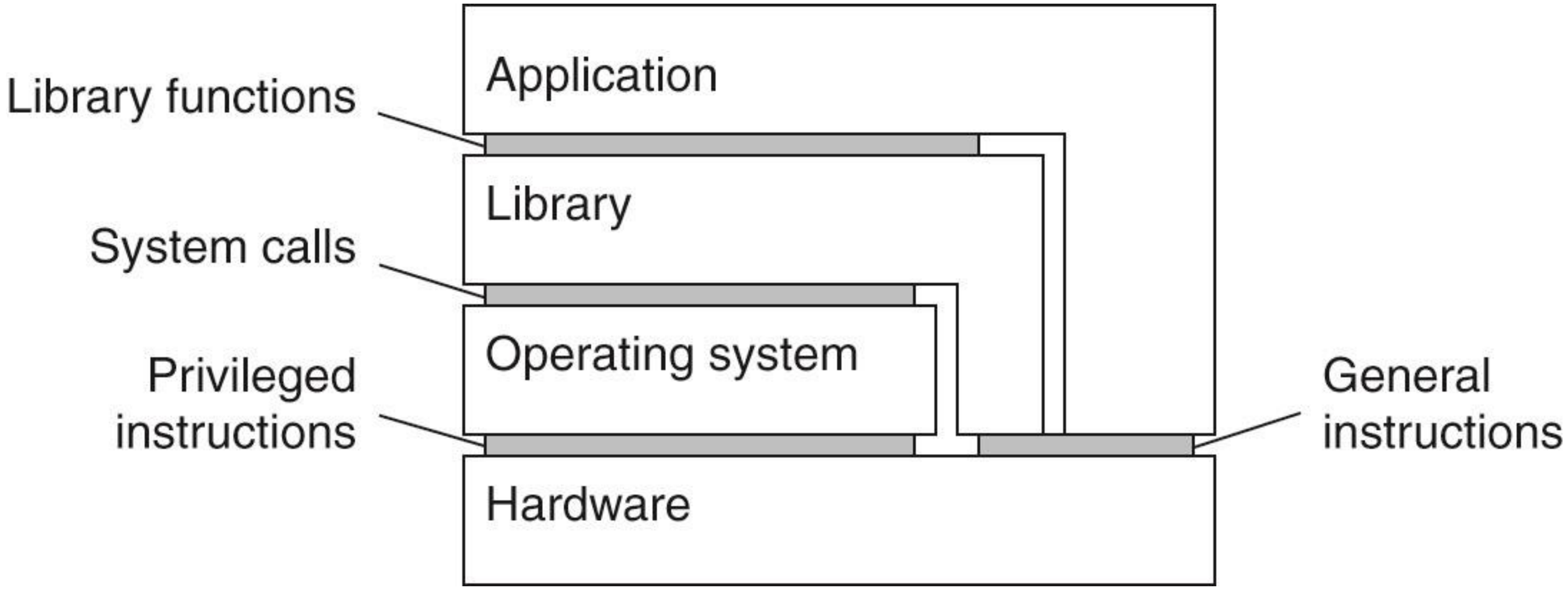

Program
Interface A
Implementation of mimicking A on B
Interface B
Hardware/software system B

(b) General organization of virtualizing system A on top of system B.

Role of Virtualization

- Helps with scalability and better utilization of hardware resources.
- Allows legacy software to run on expensive mainframe hardware.
  - Suppose you want to run an existing application on a new computer
- Runs multiple different operating systems at the same time.
- Provides a high degree of portability and flexibility.

Architectures of Virtual Machines



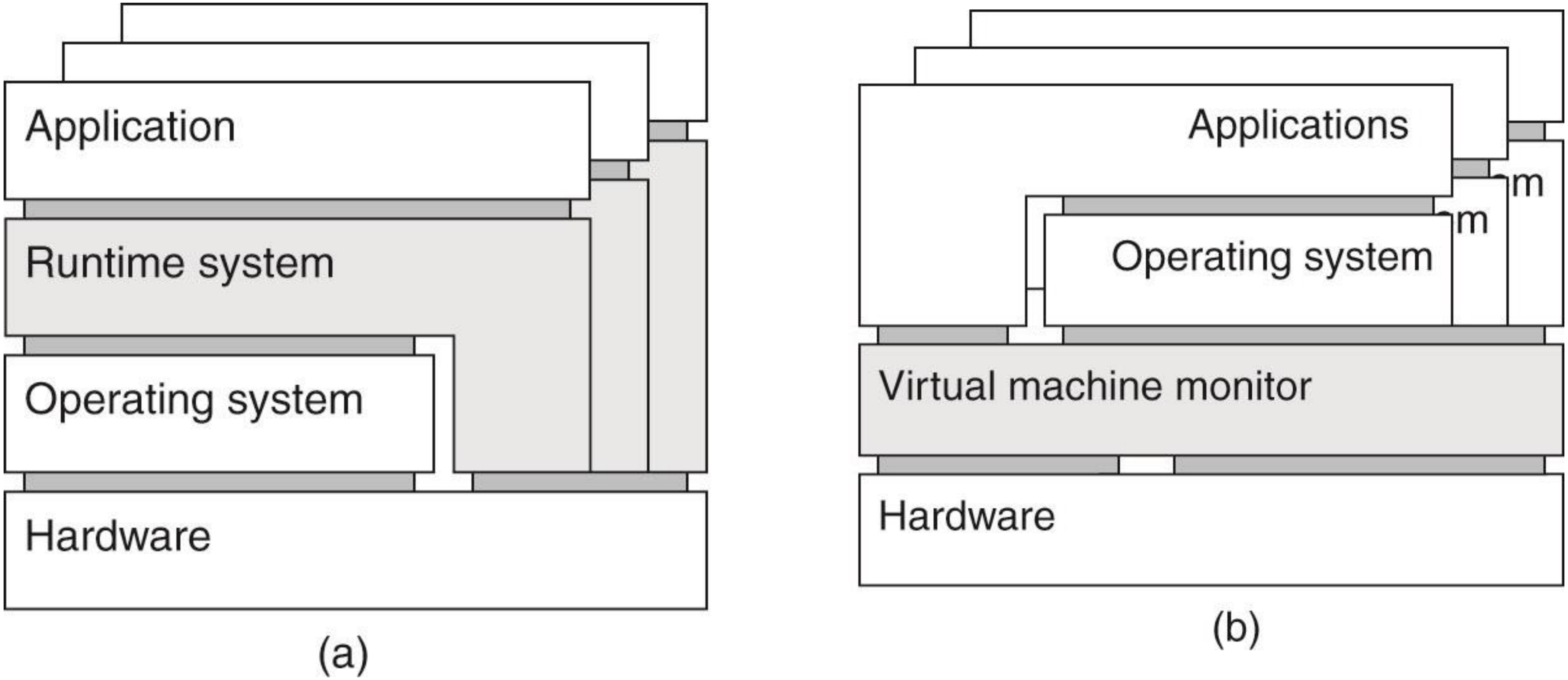
Architectures of Virtual Machines

- Four types of interfaces:
  - An interface between the hardware and software, consisting of machine instructions
    - Can be invoked by any program.
  - An interface between the hardware and software, consisting of machine instructions
    - can be invoked only by privileged programs, such as an operating system.
  - An interface consisting of system calls as offered by an operating system.
  - An interface consisting of library calls,
    - known as an application programming interface (API).

Architectures of Virtual Machines

- Virtualization can be implemented at two levels.
  - **Process Virtual Machine:**
    - Virtualization is done essentially only for executing a single process (program).
    - An abstract instruction set that is to be used for executing applications.
    - For example: Java runtime, Windows emulation (Wine) on Unix/Linux/MacOS.
  - **Virtual Machine Monitor:**
    - Composed of the host OS and the virtualization software.
    - A layer completely shielding the original hardware but offering the complete instruction set of that same (or other hardware) as an interface.
    - Provides a further decoupling between hardware and software allowing moving complete environment from one machine to another.
    - Makes it possible to have multiple instances of different operating systems run simultaneously and concurrently on the same platform.
    - Examples: Vmware, VirtualBox, Xen, VirtualPC, Parallels etc.

Architectures of Virtual Machines



A process virtual machine, with multiple instances of (application, runtime) combinations. (a)
(b) A virtual machine monitor, with multiple instances of (applications, OS) combinations.

Servers

- There are several ways to organize servers:
  - In the case of an iterative server, the server itself handles the request and, if necessary, returns a response to the requesting client.
  - A concurrent server can be multi-threaded or multi-process. It does not handle the request itself, but passes it to a separate thread or another process, after which it immediately waits for the next incoming request.
- A server can be stateless or stateful.
  - A stateless server does not remember anything from one request to another. For example, a HTTP server is stateless.
  - Stateful servers maintains information about its clients.

Design Issues for Servers

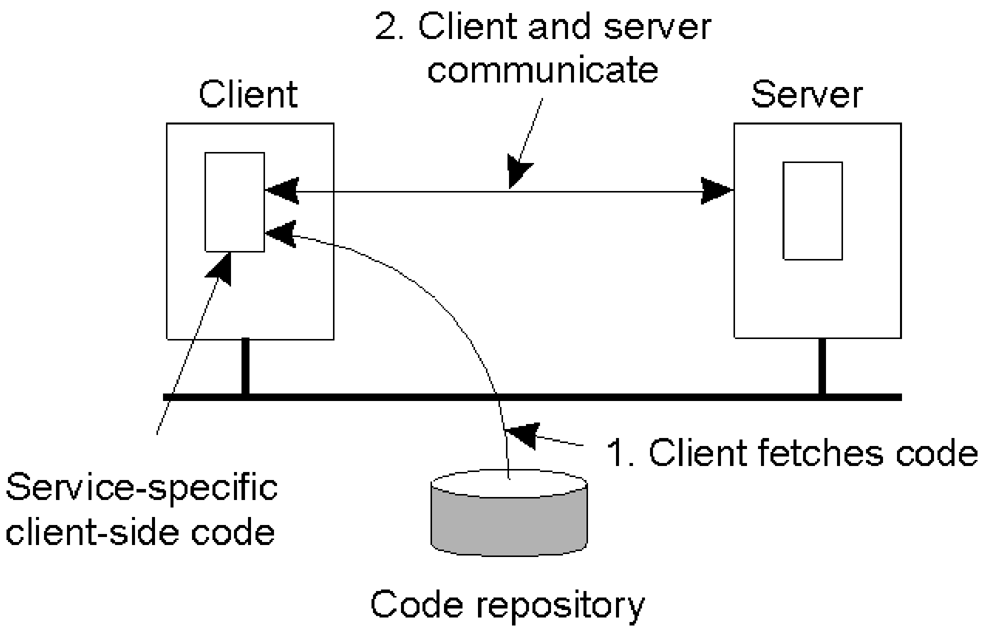
- Where clients contact a server?
  - In all cases, clients send requests to an end point, also called a port, at the machine where the server is running.
  - Each server listens to a specific end point:
    - Servers that handle Internet FTP requests always listen to TCP **port 21**.
    - An HTTP server for the www listen to TCP port 80.
- How to handle communication interrupts?
  - Use out-of-band data. Example: to cancel the upload of a huge file.
  - Server listens to separate endpoint, which has higher priority, while also listening to the normal endpoint (with lower priority).
  - Send urgent data on the same connection. Can be done with TCP, where the server gets a signal (SIGURG) on receiving urgent data.

Reasons for Code Migration

- Improve computing performance by moving processes from heavily-loaded machines to lightly loaded machines.
  - E.g. Java applets
- Improve communication times by shipping code to systems where large data sets reside.
  - E.g. a client ships code to a database server or vice versa.

Reasons for Code Migration

- Flexibility to dynamically configure distributed systems.

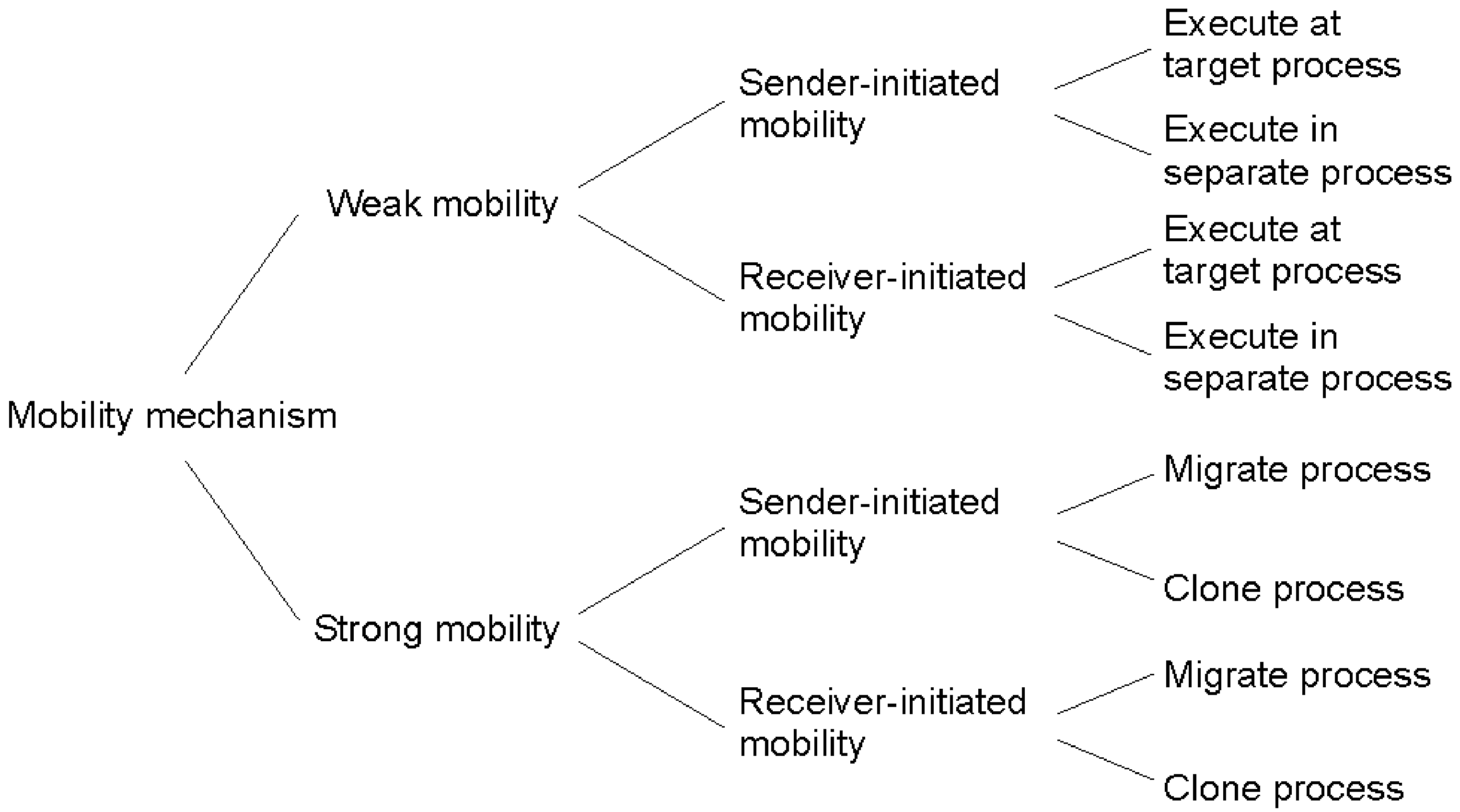

2. Client and server communicate
Client
Server
1. Client fetches code
Service-specific client-side code
Code repository

The principle of dynamically configuring a client to communicate to a server. The client first fetches the necessary software, and then invokes the server

Migration Models

- A process consists of three segments: code segment, resource segment, execution segment.
- Migration (or mobility) can be **weak** or **strong**:
  - Weak migration: only the code segment can be transferred. Transferred program always starts at one of the predefined starting positions. E.g. java applets.
  - Strong migration: code and execution segments can both be transferred.
- Also, migration can be **sender-initiated** or **receiver-initiated**:
  - Sender initiated: uploading code to a server. Ex: Client sending a query to database server.
  - Receiver-initiated: downloading code from a server by a client. Ex: client downloading Java applets from a server.

Migration Models


Mobility mechanism
Weak mobility
Strong mobility
Sender-initiated mobility
Receiver-initiated mobility
Sender-initiated mobility
Receiver-initiated mobility
Execute at target process
Execute in separate process
Execute at target process
Execute in separate process
Migrate process
Clone process
Migrate process
Clone process

Resource Migration

- Depends on type of “resource”:
  - By identifier: specific web site, ftp server
  - By value: Java libraries
  - By type: printers, local devices
- Depends on type of “attachments”
  - Unattached to any node: data files
  - Fastened resources (can be moved only at high cost)
    - Ex: database, web sites
  - Fixed resources
    - Ex: local devices, communication end points

Resource Migration

Process-to-Resource Binding	Resource-To-Machine Binding		
	Unattached	**Fastened**	**Fixed**
By identifier	MV (or GR)	GR (or MV)	GR
By value	CP (or MV, GR)	GR (or CP)	GR
By type	RB (or MV, CP)	RB (or GR, CP)	RB (or GR)

- Actions to be taken with respect to the references to local resources when migrating code to another machine
  - **GR:** Establish a global system-wide reference
  - **MV:** Move the resource
  - **CP:** Copy the value of the resource
  - **RB:** Rebind the process to a locally available resource

Activity

What happens to a TCP port opened by a migrating process?

A. GR: Establish a global system-wide reference

B. MV: Move the resource

C. CP: Copy the value of the resource

D. RB: Rebind the process to a locally available resource

What happen to a reference to a File when the code is moved?

A. GR: Establish a global system-wide reference

B. MV: Move the resource

C. CP: Copy the value of the resource

D. RB: Rebind the process to a locally available resource

Next Chapter

- Communication

Questions?!

Please rate your understanding of the chapter

I understand most of the chapter topics (Threads, Virtualization, Server, Code Migration)

A. Strongly Agree

B. Agree

C. Disagree

D. Strongly Disagree