Operating Systems

Comprehensive Study Guide

Chapters 15 · 16 · 17

Ch.15 — Security	Ch.16 — Virtual Machines	Ch.17 — Distributed Systems
• 40 Terms • 6 Comparisons • 6 Enum Lists	• 29 Terms • 5 Comparisons • 7 Enum Lists	• 35 Terms • 6 Comparisons • 8 Enum Lists

Terms · Comparisons · Enumeration Lists · Comprehensive Coverage

Chapter 15
Security

Threats · Attacks · Cryptography · Authentication · Firewalls · Classifications

Section 1 — Terminology & Definitions

Term	Definition / Details
Security	A system is considered secure if its resources are used and accessed only as intended under all circumstances. Absolute security is unachievable in practice.
Intruder / Cracker	An entity that attempts to breach the security of a system. Distinct from an authorized user.
Threat	A potential security violation — a condition or situation that could lead to a breach of security.
Attack	An actual attempt to breach security. Can be accidental or malicious. It is easier to protect against accidental than malicious misuse.
Breach of Confidentiality	Unauthorized reading of data. Sensitive information is accessed by parties not permitted to view it.
Breach of Integrity	Unauthorized modification of data. Data is altered by an unauthorized party without the knowledge or consent of the owner.
Breach of Availability	Unauthorized destruction of data, making it unavailable to legitimate users.
Theft of Service	Unauthorized use of resources — an attacker uses computing resources (CPU, bandwidth, storage) without permission.
Denial of Service (DOS)	Prevention of legitimate use by overloading or disabling the system. Types: Accidental (e.g., CS students writing bad fork() code) or Purposeful (extortion, punishment). Distributed DOS (DDOS) comes from multiple sites simultaneously.
Masquerading	Pretending to be an authorized user in order to escalate privileges and gain unauthorized access. This attack breaches authentication.
Replay Attack	Capturing and reusing a valid message (as-is or with modification) to trick the system into granting unauthorized access or performing unauthorized operations.
Man-in-the-Middle	An intruder intercepts the data flow, masquerading as the

Term	Definition / Details
Attack	sender to the receiver and as the receiver to the sender. Both parties believe they are communicating directly with each other.
Session Hijacking	Intercepting an already-established session to bypass authentication. The attacker takes over a valid session rather than initiating a new login.
Trojan Horse	A code segment that misuses its environment by exploiting mechanisms for allowing programs written by users to be executed by other users. May include spyware, pop-up browser windows, covert channels. Up to 80% of spam is delivered by spyware-infected systems.
Trap Door	A specific user identifier or password that circumvents normal security procedures. Could be included in a compiler. Detection is very difficult.
Logic Bomb	A program that initiates a security incident only when certain predefined circumstances or conditions are met (e.g., a specific date or user action).
Stack / Buffer Overflow	Exploits a bug in a program by overflowing the stack or memory buffers due to failure to check bounds on inputs or arguments. The attacker writes past arguments on the stack into the return address, redirecting execution to malicious code pointed to on the stack. This can cause unauthorized user access or privilege escalation.
Virus	A code fragment embedded in a legitimate program that is self-replicating and designed to infect other computers. Very specific to CPU architecture, OS, and applications. Usually borne via email or macro. A virus dropper inserts viruses onto the system. Categories: file/parasitic, boot/memory, macro, source code, polymorphic (encrypted, stealth, tunneling, multipartite, armored).
Worm	Malware that uses a spawn mechanism and is a standalone program. The famous Internet worm exploited UNIX networking features (remote access) and bugs in finger and sendmail programs, and used trust-relationship mechanisms of rsh to access systems without a password.
Why Windows is Targeted	Most common OS. Everyone runs as an administrator. Licensing issues. Monoculture considered harmful — widespread uniformity increases attack surface.
Botnet	A network of compromised ("zombie") computers controlled by attackers, used for spam delivery, DDOS attacks, and keystroke logging to steal passwords and credit card numbers.
Port Scanning	Automated attempt to connect to a range of ports on one or more IP addresses. Detects answering services, OS version, and running applications. Tools: nmap (scans all ports in a given IP range) and nessus (has database of protocols, bugs, exploits). Frequently launched from zombie systems to decrease traceability.

Term	Definition / Details
Cryptography	The broadest security tool available. Used to constrain potential senders/receivers of messages based on secrets (keys). Network message source/destination cannot be trusted without it. Enables: confirmation of source, receipt only by intended destination, trust relationship between sender and receiver.
Encryption Algorithm	Consists of: Set K of keys, Set M of messages, Set C of ciphertexts, function E:K→(M→C) for encryption, function D:K→(C→M) for decryption. Essential property: given ciphertext c, a computer can compute m such that E(k)(m)=c ONLY if it possesses D(k). Without D(k), ciphertexts cannot be decrypted.
Symmetric Cryptography	Same key used for both encryption and decryption. Based on transformations. Less computationally intensive than asymmetric. Used for bulk data encryption.
Asymmetric Cryptography	Different keys for encryption and decryption (public/private key pairs). Based on mathematical functions. Much more compute-intensive. Typically NOT used for bulk data encryption.
Authentication Algorithm	Constrains the set of potential senders of a message. Complementary to encryption; also proves message was not modified. Components: Set K of keys, Set M of messages, Set A of authenticators, function S:K→(M→A) to generate authenticators, function V:K→(M×A→{true,false}) to verify. A computer not holding S(k) cannot generate valid authenticators.
Certificate / Certificate Authority (CA)	A Certificate is proof of who owns a public key — a public key digitally signed by a trusted party. The Certificate Authority (CA) receives proof of identification and certifies the public key belongs to an entity. CA public keys are included with web browser distributions. CAs vouch for other authorities by signing their keys.
Passwords	Most common user authentication method; can be considered a special case of keys or capabilities. Rules: must be kept secret, changed frequently, use history to avoid repeats, be non-guessable, log invalid access attempts (not the passwords themselves). Risk: shoulder surfing, Trojan horse keystroke loggers, sniffing.
Password Storage (Hashing + Salt)	Encrypt passwords to avoid keeping them secret; still keep file secret (Unix: /etc/shadow, superuser-only readable). Use algorithm easy to compute but hard to invert. Only encrypted password stored, never decrypted. Add "salt" to avoid the same password encrypting to the same value, defeating precomputed hash attacks.
One-Time Passwords	Passwords computed using a function based on a seed, known to both user and authenticating computer. Generated by hardware devices, calculators, or key fobs. Changes very frequently. Defeats replay attacks and sniffing.

Term	Definition / Details
Biometrics	Authentication using a physical attribute of the user, such as a fingerprint or hand scan.
Multi-Factor Authentication	Requires two or more independent authentication factors. Example: USB "dongle" (something you have) + biometric measure (something you are) + password (something you know).
Defense in Depth	The most common security theory: multiple layers of security are applied so that if one layer fails, others remain. Requires a security policy, vulnerability assessment, and intrusion detection.
Security Policy	A document describing what is being secured and the rules for protecting system resources.
Vulnerability Assessment	Compares the real state of a system or network against the security policy to identify weaknesses and gaps.
Intrusion Detection	Detects attempted or successful intrusions. Signature-based: spots known bad patterns. Anomaly-based: spots differences from normal behavior — can detect zero-day attacks. Both suffer from false positives and false negatives. Includes virus protection.
Firewall	Placed between trusted and untrusted hosts to limit network access. Can be tunneled (disallowed protocol carried inside allowed protocol, e.g., telnet inside HTTP) or spoofed (rules based on hostname/IP which can be faked). Types: personal (software on host), application proxy (understands application protocol, e.g., SMTP), system-call (monitors all important system calls).
DMZ (Demilitarized Zone)	A network segment separated from both the internal company network and the internet by firewalls. Provides a buffer zone for internet-accessible services while protecting the internal network.
Computer Security Divisions (DoD)	D: Minimal security. C1: cooperating users same protection level, discretionary protection via auditing. C2: user-level access control. B: all C properties + unique sensitivity labels per object (B1, B2, B3). A: formal design and verification techniques.
Auditing, Accounting, Logging	Recording all or specific system or network activities for security review, compliance, and forensic analysis.

⚖ Section 2 — Comparisons & Differentiations

Types of Security Breaches

Breach Type	What Is Violated	Description
Confidentiality	Secrecy of data	Unauthorized reading of data
Integrity	Correctness of data	Unauthorized modification of data
Availability	Access to data	Unauthorized destruction of data
Theft of Service	Use of resources	Unauthorized use of computing resources
Denial of Service (DOS)	Access by legitimate users	Prevention of legitimate use by overloading the system

Types of Security Attacks (Attack Methods)

Attack	How It Works	Key Characteristic
Masquerading	Pretends to be an authorized user	Breaches authentication — privilege escalation
Replay Attack	Reuses captured valid messages (as-is or modified)	Uses legitimate past data against the system
Man-in-the-Middle	Intercepts flow; impersonates both sender and receiver	Both parties unaware of interception
Session Hijacking	Takes over an existing established session	Bypasses authentication entirely

Program Threats Comparison

Threat	Description	Propagation / Trigger
Trojan Horse	Misuses environment; hidden in legitimate program	Downloads, email; runs when parent program runs
Trap Door	Secret bypass ID/password in code	Built into compiler or application
Logic Bomb	Security incident under specific conditions	Triggered by event, date, or condition
Buffer Overflow	Exploits missing bounds check to redirect execution	Malformed input overflows stack/buffer
Virus	Self-replicating code fragment in legitimate	Email, macros; infects other files

Threat	Description	Propagation / Trigger
	program	
Worm	Standalone self-replicating program	Network vulnerabilities; no host program needed

Symmetric vs. Asymmetric Cryptography

Aspect	Symmetric	Asymmetric
Keys	Same key for encrypt and decrypt	Different keys: public (encrypt) and private (decrypt)
Basis	Transformations	Mathematical functions
Compute cost	Low — fast	High — much more compute-intensive
Use case	Bulk data encryption	Key exchange, digital signatures, authentication
Bulk encryption	Yes	No — typically not used for bulk data

Types of Firewalls

Firewall Type	How It Works	Level / Scope
Network Firewall	Placed between trusted and untrusted hosts; limits network access based on rules	Network perimeter
Personal Firewall	Software layer on a single host; monitors/limits traffic to and from that host	Single machine
Application Proxy Firewall	Understands and controls application-layer protocols (e.g., SMTP, HTTP)	Application layer
System-Call Firewall	Monitors important system calls; applies rules (e.g., this program can execute that syscall)	OS/kernel level

DoD Computer Security Divisions (A–D)

Division	Subdivisions	Description
D	None	Minimal security — no

Division	Subdivisions	Description
		meaningful protection requirements
C	C1, C2	Discretionary protection via auditing. C1: cooperating users same protection. C2: user-level access control
B	B1, B2, B3	All C properties PLUS each object may have unique sensitivity labels
A	None	Uses formal design and verification techniques to ensure security

Section 3 — Enumeration Questions

4 Levels at Which Security Must Occur

4 Levels at Which Security Must Occur
1. **Physical** — Data centers, servers, connected terminals — physical access control.
2. **Human** — Avoid social engineering, phishing, dumpster diving — the human element.
3. **Operating System** — Protection mechanisms, debugging, process/memory isolation.
4. **Network** — Intercepted communications, interruption, denial of service.

Categories of Viruses

Categories of Viruses
1. **File / Parasitic** — Attaches itself to executable files and activates when the file runs.
2. **Boot / Memory** — Infects the boot sector or resides in memory, active before the OS loads.
3. **Macro** — Written in macro languages embedded in documents (e.g., Microsoft Word macros).
4. **Source Code** — Modifies source code files before compilation.
5. **Polymorphic** — Changes form to avoid signature detection. Sub-types: Encrypted, Stealth, Tunneling, Multipartite, Armored.

Encryption Algorithm Components

Encryption Algorithm Components
1. Set K of keys — All possible encryption/decryption keys.
2. Set M of messages — All possible plaintext messages.
3. Set C of ciphertexts — All possible encrypted messages.
4. Function E: K→(M→C) — For each key k, E(k) generates ciphertext from message. Must be efficiently computable.
5. Function D: K→(C→M) — For each key k, D(k) recovers message from ciphertext. Must be efficiently computable.

Authentication Algorithm Components

Authentication Algorithm Components
1. Set K of keys — All possible authentication keys.
2. Set M of messages — All possible messages to be authenticated.
3. Set A of authenticators — All possible authentication tags/signatures.
4. Function S: K→(M→A) — Signing function — for each key k, S(k) generates authenticator from message.
5. Function V: K→(M×A→{true,false}) — Verification function — V(k) verifies that authenticator is valid for a message.

User Authentication Methods

User Authentication Methods
1. Passwords — Most common method. Must be non-guessable, changed frequently, with history tracking. Log failed attempts.
2. Hashed Passwords + Salt — One-way hash (easy to compute, hard to invert). Salt prevents same password → same hash. Never decrypted. Stored in /etc/shadow.
3. One-Time Passwords — Seed-based computation known by both sides. Generated by hardware devices/key fobs. Changes very frequently.
4. Biometrics — Physical attribute such as fingerprint or hand scan.
5. Multi-Factor Authentication — Two or more independent factors required (something you have + something you are + something you know).

Security Defense Strategies (Defense in Depth)

Security Defense Strategies (Defense in Depth)

1. **Security Policy** — Describes what is being secured.
2. **Vulnerability Assessment** — Compares real system state vs. security policy.
3. **Intrusion Detection** — Signature-based (known patterns) or anomaly-based (deviations from normal). Can detect zero-day attacks.
4. **Virus Protection** — Antivirus software to detect and quarantine malicious code.
5. **Auditing, Accounting, and Logging** — Records all or specific system/network activities.
6. **Firewalling** — Multiple firewall types (personal, application proxy, system-call, network).

Chapter 16
Virtual Machines

Hypervisors · Paravirtualization · Memory · I/O · Storage · Live Migration

Section 1 — Terminology & Definitions

Term	Definition / Details
Virtual Machine (VM)	An abstraction of the hardware of a single computer into several different execution environments. Similar to a layered approach but the layer creates a virtual system on which OSes or applications can run.
Host	The underlying physical hardware system on which the VMM (hypervisor) runs and manages virtual machines.
Guest	A process (usually an OS) provided with a virtual copy of the host hardware by the VMM. A single physical machine can run multiple guest OSes concurrently.
VMM / Hypervisor	Virtual Machine Manager — creates and runs virtual machines by providing an interface identical (or near-identical in paravirtualization) to the host hardware. Must be in complete control of system resources.
Type 0 Hypervisor	Hardware-based solution implemented in firmware (not software). Old idea known by names "partitions" or "domains". OS needs nothing special; VMM is in firmware. Smaller feature set. Each guest has dedicated HW. I/O is challenging. Can provide virtualization-within-virtualization. Examples: IBM LPARs, Oracle LDOMs.
Type 1 Hypervisor (Dedicated)	Special-purpose OS that runs natively on hardware (bare metal), in kernel mode. Rather than providing system call interface, it creates and manages guest OSes. Implements device drivers for host HW. Guests generally do not know they are in a VM. Can run on Type 0 but NOT on another Type 1. Found in datacenters. Examples: VMware ESX, Joyent SmartOS, Citrix XenServer.
Type 1 Hypervisor (General OS)	A general-purpose OS that also provides VMM functionality in addition to normal duties. Treats guests as just another process with special handling for privileged instructions. Typically less feature-rich than dedicated Type 1. Examples: RedHat Linux with KVM, Windows with Hyper-V, Oracle Solaris.
Type 2 Hypervisor	Application running on a standard host OS providing VMM features to guest OSes. VMM is simply another process managed by the host. Even host does not know a VMM is

Term	Definition / Details
	running guests. Poorer overall performance (cannot take advantage of some HW features) but requires NO changes to host OS. Examples: VMware Workstation/Fusion, Parallels Desktop, Oracle VirtualBox.
Paravirtualization	Does NOT fit the definition of full virtualization — VMM does not present an exact hardware duplicate. Guest OS must be modified to use VMM services. Leads to increased performance (direct cooperation). Less needed as hardware VM support grows. Xen is the leader: uses clean device abstractions, circular shared buffers, hypercalls instead of system calls for page table changes. On modern CPUs Xen no longer requires guest modification → no longer truly paravirtualization.
Hypercall	A call from a paravirtualized guest OS directly to the hypervisor to request services, analogous to a system call from user space to the OS kernel. Used for page-table changes in Xen.
Programming-Environment Virtualization	VMMs that do not virtualize real hardware but create an optimized virtual environment for a programming language. Defined as providing APIs that define features made available to a language. JVM compiled to run on many systems ("write once, run anywhere"). Similar to interpreted languages. Examples: Oracle Java (JVM), Microsoft .NET.
JVM (Java Virtual Machine)	Invented by Sun Microsystems in 1995. Java objects compiled to architecture-neutral bytecode (.class files). JVM class loader loads bytecodes; JVM compiled per architecture reads and executes them. Includes garbage collection. JIT (Just-In-Time) compiler turns bytecodes into native code and caches them for speed. JVM compiled to run on many systems including smartphones.
Emulator	Older approach allowing one OS to run on a completely different CPU architecture. ALL guest instructions must be translated from guest CPU to native CPU — true emulation, not virtualization. Performance is orders of magnitude slower than native (new faster machines reduce this gap). Very popular in gaming (old consoles emulated on new hardware).
Application Containment	Not true virtualization but uses similar techniques. Segregates applications from the OS making them more secure/manageable. Only ONE kernel runs (host OS). Creates virtual layer between OS and apps. Each container/zone has own apps, networking stack, addresses, ports, user accounts. CPU and memory resources divided between zones. Examples: Oracle Solaris Zones, BSD Jails, IBM AIX WPARs.
VCPU (Virtual CPU)	A virtual representation of CPU state per guest as the guest believes it to be. When a guest is context-switched onto the physical CPU by the VMM, VCPU information is loaded and stored. Can be configured per guest and adjusted throughout the VM lifecycle.
CPU	Situation where total VCPUs configured across all guests

Term	Definition / Details
Overcommitment	exceeds available physical CPUs. VMM uses standard scheduling algorithms (+ fairness aspect) to share CPU time. Results in cycle stealing and guests receiving fewer/less predictable CPU cycles than expected.
Cycle Stealing	When the VMM takes CPU cycles from guests to do its own work. Combined with overcommitment, causes guests to receive fewer cycles than expected, leading to poor response times and incorrect time-of-day clocks.
VMware ESX Memory Management (3 Methods)	1. Double-paging: guest page table says page is in physical frame but VMM has moved it to backing store. 2. Balloon memory manager: pseudo-device driver installed in guest communicates with VMM, told to allocate/deallocate memory — causes guest OS to free or gain physical memory. 3. Deduplication: VMM detects same page loaded in multiple guests, maps all to single physical page.
Double-Paging	Guest page table indicates a page is in a physical frame, but VMM has secretly moved it to backing store. Creates extra overhead but enables memory overcommitment.
Balloon Memory Manager	A pseudo-device driver installed in each guest by the VMM. Appears as a device driver to the guest kernel but adds kernel-mode code. Communicates with VMM and is told to allocate or deallocate memory — causing the guest OS to free or gain physical memory. Used to finely control memory oversubscription.
Deduplication	VMM determines if the same page is loaded in more than one guest. Memory-maps the same physical page into multiple guests, saving physical memory. Effective when guests run same OS/applications.
I/O in VMMs	I/O is complicated for VMMs. Many short fast paths in standard OSes for improved performance. Less the hypervisor needs to do for I/O, the better. Options: direct device access, DMA pass-through, direct interrupt delivery — all require HW support. Networking: VMM can bridge guest to network (direct access) and/or provide NAT (address translation).
Storage in VMMs	VMM must provide both boot disk and general data access. Standard disk partitioning not sufficient for dozens of guests. Type 1: stores guest root disks/config within VMM file system as disk images. Type 2: stores as files in host OS file system. Duplicate file → create new guest. Move file → move guest. P-to-V and V-to-P conversions available.
Live Migration	Move a running VM from one physical host to another without interrupting user access. Very useful for resource management and maintenance. Steps: (1) source VMM connects to target, (2) target creates new guest/VCPU, (3) source sends read-only pages, (4) source sends read-write pages (marked clean), (5) repeats for dirty pages, (6) source freezes guest, sends final VCPU state + dirty pages, tells target to start, (7) once target confirms guest running, source terminates its copy.

Term	Definition / Details
VM Lifecycle	Created by VMM → resources assigned (cores, memory, networking, storage details) → runs → when no longer needed, deleted (freeing resources). Steps simpler and faster than physical machine installation. Can lead to VM sprawl if many VMs accumulate without tracking.
VM Sprawl	Proliferation of many VMs whose history and state become difficult to track, because creating VMs is much simpler than physical machine installation.
Snapshot	A saved state of a VM at a specific point in time. VM can be restored back to that state. Some VMMs allow multiple snapshots per VM. Enables freezing, suspending, copying, and moving VMs.
Templating	Creating an OS + application VM combination and using it to rapidly deploy multiple identical instances for customers or deployment environments.
Cloud Computing	Built on all VM features combined (snapshots, cloning, live migration, templating). Programs use APIs to tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, and virtual desktops on demand.

⚖ Section 2 — Comparisons & Differentiations

Type 0 vs. Type 1 (Dedicated) vs. Type 1 (General) vs. Type 2 Hypervisors

Aspect	Type 0	Type 1 Dedicated	Type 1 General OS	Type 2
Implementation	Firmware	Bare metal (kernel mode)	General-purpose OS	Application on host OS
Guest modification needed	No	No	No	No
Performance	Dedicated HW per guest	Highest	Slightly less than T1 dedicated	Lowest (can't use some HW features)
Primary use	Mainframe / enterprise	Datacenter	General server	Desktop / development / students
Examples	IBM LPARs, Oracle LDOMs	VMware ESX, Citrix XenServer	RHEL+KVM, Windows+Hyper-V	VMware Workstation, VirtualBox
Can run on Type 0?	N/A	Yes	Yes	Yes

Aspect	Type 0	Type 1 Dedicated	Type 1 General OS	Type 2
Can run on Type 1?	N/A	NO	NO	Yes (on host OS running on T1)
Host OS required	No (firmware)	No	Yes (it IS the OS)	Yes
Knows guests are VMs?	Guests usually unaware	Guests generally unaware	Guests treated as processes	Host unaware VMM is running

Full Virtualization vs. Paravirtualization

Aspect	Full Virtualization	Paravirtualization
Hardware presented	EXACT hardware duplicate	Does NOT present exact hardware
Guest modification	Guest needs NO modification	Guest MUST be modified to use VMM services
Guest-VMM communication	Standard system calls	Hypercalls directly to hypervisor
Performance	Standard	Better — direct cooperation between guest and VMM
Disadvantage	Less efficient I/O communication	Requires guest OS source code access and modification
Modern trend	Standard approach	Less needed as HW VM support grows; Xen now runs without modification on modern CPUs

Virtualization vs. Emulation vs. Containment

Aspect	Virtualization	Emulation	Containment
CPU requirement	Same CPU architecture as host	Different CPU arch allowed — translates all instructions	Same CPU as host
Performance	Near-native	Order of magnitude slower	Near-native (less overhead)
Kernels running	One per guest	One per guest	ONE shared kernel (host OS)
Guest isolation	Full (separate	Full	Partial (shared

Aspect	Virtualization	Emulation	Containment
	kernels)		kernel, isolated app environment)
Use case	Multiple OS environments	Legacy apps, gaming	App security/management without full VM overhead
Examples	VMware, KVM	QEMU, gaming emulators	Solaris Zones, BSD Jails, Docker containers

VMware ESX 3 Memory Management Methods

Method	How It Works	Effect on Guest
Double-Paging	Guest page table says page is in physical frame; VMM secretly moves it to backing store	Guest unaware; extra paging overhead
Balloon Memory Manager	Pseudo-driver in guest communicates with VMM; told to allocate/deallocate memory	Guest OS frees or gains physical memory as directed
Deduplication	VMM maps identical pages from multiple guests to single physical page	Transparent to guests; reduces total physical memory usage

Xen Paravirtualization Techniques

Technique	Description
Clean device abstractions	Simple and efficient device interfaces between guest and VMM
Circular shared buffers	Each device has a circular buffer shared by guest and VMM via shared memory for efficient I/O
Hypercalls	Guest uses hypercalls instead of system calls when page-table changes needed
Read-only page tables	Memory management does not include nested page tables; each guest has own read-only tables
Modern CPUs	Xen no longer requires guest modification on modern CPUs → no longer truly paravirtualization

Section 3 — Enumeration Questions

3 Formal Conditions for Virtualization (1972 Definition)

3 Formal Conditions for Virtualization (1972 Definition)
1. Identical Environment — A VMM provides an environment essentially identical to the original machine.
2. Minor Performance Decrease — Programs running within that environment show only minor performance decreases.
3. Complete Resource Control — The VMM is in complete control of system resources.

8 Types of Hypervisors and Virtualization Approaches

8 Types of Hypervisors and Virtualization Approaches
1. Type 0 Hypervisor — Hardware-based, firmware implementation, dedicated HW per guest. Examples: IBM LPARs, Oracle LDOMs.
2. Type 1 Hypervisor (Dedicated) — Bare metal OS, kernel mode, highest performance. Examples: VMware ESX, Citrix XenServer.
3. Type 1 Hypervisor (General OS) — General OS + VMM functions. Examples: RHEL+KVM, Windows+Hyper-V.
4. Type 2 Hypervisor — Runs as user process on host OS, lowest performance, no host changes needed. Examples: VMware Workstation, VirtualBox.
5. Paravirtualization — Guest modified to use VMM services via hypercalls. Better performance. Xen is the leader.
6. Programming-Environment Virtualization — Virtual environment for a programming language (JVM for Java, .NET runtime).
7. Emulators — Translates ALL guest instructions to different native CPU architecture. Order of magnitude slower.
8. Application Containment — Segregates apps from OS. Single shared kernel. Examples: Solaris Zones, BSD Jails, IBM AIX WPARs.

Benefits and Features of Virtual Machines

Benefits and Features of Virtual Machines
1. Isolation / Protection — Host protected from VMs; VMs protected from each other. Virus less likely to spread. Sharing available via shared filesystem or network.
2. Snapshots — Freeze and save VM state; restore to any saved snapshot. Multiple snapshots per VM. Enables freeze/suspend/resume and move/copy.
3. Cloning — Create a copy of VM and run both original and copy simultaneously.

4. **Templating** — Create OS + application VM template and deploy multiple identical instances.
5. **Live Migration** — Move running VM between hosts with no interruption to users.
6. **OS Research & Dev Efficiency** — Run multiple different OSes on single machine for research, consolidation, and app development.
7. **Cloud Computing** — All features combined enable dynamic provisioning via APIs.

OS Component Challenges from Virtualization

OS Component Challenges from Virtualization
1. **CPU Scheduling** — Even single-CPU systems act like multiprocessor. CPU overcommitment forces VMM scheduling. Cycle stealing causes poor timeslice accuracy and incorrect time-of-day clocks. VMMs provide fix-up applications in guests.
2. **Memory Management** — Oversubscription requires double-paging, balloon manager, or deduplication. VMware ESX uses all 3 methods.
3. **I/O** — Many short fast paths in standard OSes. Less hypervisor does, the better. Options: direct device access, DMA pass-through, direct interrupt delivery (all need HW support).
4. **Storage** — Must support dozens of guests. Type 1: disk images in VMM filesystem. Type 2: files in host OS filesystem. P-to-V and V-to-P conversions.
5. **Networking** — VMM and guests all need network access. VMM can bridge to network and/or provide NAT.

7 Steps of Live Migration

7 Steps of Live Migration
1. **Step 1** — Source VMM establishes a connection with the target VMM.
2. **Step 2** — Target creates a new guest by creating a new VCPU and allocating resources.
3. **Step 3** — Source sends all read-only guest memory pages to the target.
4. **Step 4** — Source sends all read-write pages to the target, marking them as clean.
5. **Step 5** — Source repeats step 4 for pages that became dirty during the transfer.
6. **Step 6** — When dirty-page cycle becomes very short: source VMM freezes guest, sends VCPU final state + other state + final dirty pages, tells target to start the guest.
7. **Step 7** — Once target confirms guest is running, source terminates its copy of the guest.

VMware Workstation Features (Type 2 Example)

VMware Workstation Features (Type 2 Example)
1. Platform — Runs on x86, provides VMM for guest OSes. Type 2 — application on host OS.
2. Guest variety — Windows, Linux, etc., all runnable concurrently (as resources allow).
3. Virtualization layer — Abstracts underlying HW, providing each guest its own virtual CPUs, memory, disk drives, network interfaces.
4. Storage options — Physical disks provided directly to guests, or virtual disks (just files within host file system).

Java Virtual Machine (JVM) Key Features

Java Virtual Machine (JVM) Key Features
1. Write once, run anywhere — Java programs compiled to architecture-neutral bytecode (.class files) run on any system with a JVM.
2. JVM compiled per architecture — JVM reads bytecode and executes; runs on many systems including smartphones.
3. Class loader — JVM class loader loads .class files (Java objects/programs) into the JVM for execution.
4. Garbage collection — Automatically reclaims memory no longer in use — no manual memory management.
5. JIT compiler — Just-In-Time compiler turns bytecodes into native code and caches them for improved performance.

Chapter 17
Distributed Systems

Networks · Communication · Protocols · Robustness · DFS

Section 1 — Terminology & Definitions

Term	Definition / Details
Distributed System	A collection of loosely coupled processors interconnected by a communications network. Processors are called nodes, computers, machines, or hosts. Allows resource sharing, computation speedup, reliability, and communication. Can downsize computers with more flexibility and better user interfaces.
Site	The physical location of a processor in a distributed system. Generally a server at one site has a resource that a client at a different site wants to use.
Server	A machine at a site that provides resources or services to a priori unknown clients. In DFS context: service software running on a single machine.
Client	A process or machine that invokes a service using a set of operations forming its client interface. In DFS: accesses primitive file operations (create, delete, read, write) provided by the server.
Network Operating System	Users ARE aware of the multiplicity of machines. Remote resource access is explicit: remote login (telnet, ssh, Remote Desktop), or file transfer (FTP). Users must change paradigms — establish sessions, give network-based commands. More difficult for users.
Distributed Operating System	Users are NOT aware of the multiplicity of machines. Access to remote resources is similar to local access. Provides data migration, computation migration, and process migration transparently.
Data Migration	Transferring data between sites — either the entire file or only the portions of the file necessary for the immediate task.
Computation Migration	Transferring the computation rather than the data across the system. Achieved via Remote Procedure Calls (RPCs) or a messaging system. The computation runs where the data lives.
Remote Procedure Call (RPC)	A mechanism for computation migration — a process invokes a procedure that executes on a remote site. The caller does

Term	Definition / Details
	not need to know the procedure is executing remotely.
Process Migration	Executing an entire process or parts of it at different sites. Reasons: load balancing (even workload), computation speedup (concurrent subprocesses at different sites), hardware preference (specialized processor elsewhere), software preference (required software only at specific site), data access (run process where data is, avoiding large data transfers).
Local-Area Network (LAN)	Designed to cover a small geographical area. Multiple topologies (star, ring). Speeds: 1 Mbps (Appletalk, Bluetooth) to 40 Gbps (fastest Ethernet over copper/fiber). Components: computers, peripherals, routers. Ethernet (IEEE 802.3) is most common — multiaccess bus-based. Wireless (WiFi, IEEE 802.11g at 54 Mbps) increasingly common.
Wide-Area Network (WAN)	Links geographically separated sites via point-to-point connections over long-haul lines (often leased from phone companies). Implemented via connection processors called routers. Internet WAN enables worldwide host communication. Speeds: T1 = 1.544 Mbps; T3 = 28×T1 = 45 Mbps; OC-12 = 622 Mbps. WANs and LANs interconnect (like cell tower infrastructure).
Naming and Name Resolution	One of 4 basic network design issues: How do two processes locate each other to communicate? Maps logical names to network addresses.
Routing Strategy	One of 4 basic network design issues: How are messages sent through the network from source to destination? Determines the path.
Connection Strategy	One of 4 basic network design issues: How do two processes send a sequence of messages? Connection-oriented vs. connectionless.
Contention	One of 4 basic network design issues: The network is a shared resource — how are conflicting demands for its use resolved?
OSI Model	7-layer framework: (1) Physical — raw bit transmission; (2) Data Link — node-to-node, error detection; (3) Network — routing (IP); (4) Transport — end-to-end reliability (TCP/UDP); (5) Session — establishing/maintaining sessions; (6) Presentation — data translation/encryption/compression; (7) Application — user protocols (HTTP, DNS, Telnet, SMTP, FTP).
TCP/IP Protocol Suite	Internet foundational protocols. Application layer: HTTP, DNS, Telnet, SMTP, FTP. Transport layer: TCP, UDP. Network layer: IP. Physical/Data Link: not defined by TCP/IP (handled by hardware). Notable: Session and Presentation layers are "not defined" in TCP/IP (unlike OSI).
IP Address	Unique logical network address per host. Used for routing across networks. Must be mapped to MAC address for delivery on the same network segment via ARP.

Term	Definition / Details
MAC Address	Media Access Control — unique physical hardware address assigned to each network interface card. Used for communication on the same Ethernet segment. Mapped to/from IP addresses using ARP.
DNS (Domain Name Service)	Distributed naming service that maps human-readable domain names (e.g., www.example.com) to IP addresses.
ARP (Address Resolution Protocol)	Maps MAC addresses to IP addresses on a local Ethernet network. Broadcasts a request to all hosts on the network. If hosts are on different networks, packet is sent to a router instead. Enabled only if hosts are on the same network.
Ethernet Frame	Structure: Preamble (7 bytes, 10101010 pattern), Start of Frame Delimiter (1 byte, 10101011), Destination Address (2 or 6 bytes), Source Address (2 or 6 bytes), Length of Data Section (2 bytes), Data (0–1500 bytes), Pad/optional (0–46 bytes, frame must be >63 bytes), Frame Checksum (4 bytes, error detection).
Heartbeat Protocol	Failure detection mechanism. Sites exchange periodic "I-am-up" messages at fixed intervals. If Site A does not receive a message, it sends "Are-you-up?" — can repeat or try alternate route. If no reply: failure declared (but exact cause unknown). Site A cannot determine if Site B is down, link is down, or message was lost.
Failure Reconfiguration	When failure detected: (1) if link A→B failed: broadcast to all sites. (2) if site failed: every other site notified that services of failed site are unavailable. When site/link recovers: broadcast restoration to all sites.
DFS (Distributed File System)	Distributed implementation of classical file system model where multiple users share files and storage across a network. Manages dispersed storage devices. Overall storage is composed of different, remotely located, smaller storage spaces. Challenges: Naming/Transparency and Remote File Access. Client interface should be transparent (no distinction between local and remote files).
Service (in DFS)	A software entity running on one or more machines providing a particular type of function to a priori unknown clients. Provides a client interface formed by primitive operations (create, delete, read, write for file service).
Multilevel Mapping (in DFS)	Abstraction of a file that hides how and where on disk the file is stored. A transparent DFS hides the network location of files. For replicated files: mapping returns all replica locations while hiding the existence and location of copies.
Location Transparency	File name does not reveal the file's physical storage location. Users access files by name without knowing where they are stored on the network.
Location Independence	File name does not need to change when the file's physical storage location changes. More flexible than location transparency alone.

Term	Definition / Details
Remote File Access / Caching	Reduces network traffic by retaining recently accessed disk blocks in a local cache. When not cached: copy brought from server. Accesses performed on cached copy. One master copy on server; copies scattered in client caches. Creates the cache-consistency problem.
Cache-Consistency Problem	Challenge of keeping client-cached copies consistent with the master copy on the server. Could be called "network virtual memory." Client-initiated: client checks validity with server. Server-initiated: server tracks what each client caches and reacts to detected inconsistencies.
Transparency (Design Goal)	Distributed system should appear as a conventional centralized system to the user, hiding all distribution complexity.
Fault Tolerance (Design Goal)	Distributed system should continue to function in the face of partial failures of sites or links.
Scalability (Design Goal)	As demands increase, system should easily accept addition of new resources to accommodate increased demand. Example: Hadoop framework.
Hadoop	Open-source programming framework for processing large datasets in distributed environments. Based on Google's search indexing approach. Example of scalable distributed computing.
Cluster	A collection of semi-autonomous machines that act as a single system. Provides scalability and fault tolerance.

⚖ Section 2 — Comparisons & Differentiations

Network OS vs. Distributed OS

Aspect	Network OS	Distributed OS
User awareness	Users ARE aware of multiple machines	Users NOT aware of multiple machines
Resource access	Explicit — remote login (telnet/ssh), FTP, Remote Desktop	Transparent — same as local access
User experience	Must change paradigms; establish sessions; harder	Seamless and transparent
Data access	Transfer whole files (FTP) or explicit remote access	Data/computation migration as needed
Example methods	Telnet, SSH, FTP, Remote Desktop	RPC, process migration, data migration

LAN vs. WAN

Aspect	LAN (Local-Area Network)	WAN (Wide-Area Network)
Scope	Small area (building, campus)	Large geographic area (city, country, world)
Speed range	1 Mbps (Bluetooth) to 40 Gbps (Ethernet)	T1: 1.544 Mbps; T3: 45 Mbps; OC-12: 622 Mbps
Primary technology	Ethernet (IEEE 802.3), WiFi (802.11)	Point-to-point leased lines; routers
Connection type	Shared medium (bus, switch)	Point-to-point dedicated connections
Topology	Star, ring, bus	Mesh of routers

OSI Model vs. TCP/IP Layers

OSI Layer	OSI Name	TCP/IP Equivalent
7	Application	HTTP, DNS, Telnet, SMTP, FTP
6	Presentation	Not defined (handled by application)
5	Session	Not defined (handled by application)
4	Transport	TCP, UDP
3	Network	IP
2	Data Link	Not defined (hardware dependent)
1	Physical	Not defined (hardware dependent)

Cache Consistency Approaches in DFS

Approach	Who Initiates	How It Works
Client-initiated	Client	Client initiates a validity check; server checks whether local data is consistent with master copy
Server-initiated	Server	Server records what each client caches; when server

Approach	Who Initiates	How It Works
		detects potential inconsistency, it must react and notify clients

Location Transparency vs. Location Independence

Property	Description	Implication
Location Transparency	File name does not reveal physical storage location	Users access files without knowing where they are stored
Location Independence	File name does not change when physical location changes	Files can be moved without updating client references — stronger property

Reasons for Process Migration

Reason	Description
Load Balancing	Distribute processes across network to even the workload
Computation Speedup	Subprocesses can run concurrently on different sites
Hardware Preference	Process execution may require specialized processor available only at another site
Software Preference	Required software may be available only at a particular site
Data Access	Run process remotely rather than transfer all data locally

Section 3 — Enumeration Questions

4 Reasons for Distributed Systems

4 Reasons for Distributed Systems
1. **Resource Sharing** — Sharing and printing files at remote sites, processing in distributed databases, using remote specialized hardware.
2. **Computation Speedup** — Load sharing or job migration across nodes for faster results.
3. **Reliability** — Detect and recover from site failure, transfer function to surviving sites, reintegrate failed sites.

4. **Communication** — Message passing between distributed nodes enables collaboration and coordination.

4 Basic Communication Network Design Issues

4 Basic Communication Network Design Issues
1. **Naming and Name Resolution** — How do two processes locate each other to communicate?
2. **Routing Strategies** — How are messages sent through the network from source to destination?
3. **Connection Strategies** — How do two processes send a sequence of messages (connection-oriented vs. connectionless)?
4. **Contention** — The network is a shared resource — how are conflicting demands for its use resolved?

7 OSI Model Layers (Physical → Application)

7 OSI Model Layers (Physical → Application)
1. **Physical Layer (1)** — Transmits raw bit stream over the physical medium (cables, wireless).
2. **Data Link Layer (2)** — Node-to-node data transfer; error detection and framing (Ethernet lives here).
3. **Network Layer (3)** — Routing packets across networks — IP protocol.
4. **Transport Layer (4)** — End-to-end communication and reliability — TCP (reliable) and UDP (unreliable).
5. **Session Layer (5)** — Establishing, maintaining, and terminating sessions between applications.
6. **Presentation Layer (6)** — Data translation, encryption, and compression.
7. **Application Layer (7)** — User-facing protocols: HTTP, DNS, Telnet, SMTP, FTP.

Ethernet Frame Fields (8 Fields)

Ethernet Frame Fields (8 Fields)
1. **Preamble (7 bytes)** — Pattern 10101010 repeated — synchronizes and signals start of packet.
2. **Start of Frame Delimiter (1 byte)** — Pattern 10101011 — marks the actual beginning of frame data.
3. **Destination Address (2 or 6 bytes)** — Ethernet MAC address of destination or broadcast address.

4. **Source Address (2 or 6 bytes)** — Ethernet MAC address of the sending device.
5. **Length of Data Section (2 bytes)** — Length of the data payload in bytes.
6. **Data (0–1500 bytes)** — The actual message payload.
7. **Pad / Optional (0–46 bytes)** — Padding to ensure frame is at least 64 bytes long (minimum frame size).
8. **Frame Checksum (4 bytes)** — CRC for error detection.

4 Types of Failures in Distributed Systems

4 Types of Failures in Distributed Systems
1. **Site B is down** — The remote site has completely failed and is not running.
2. **Direct link between A and B is down** — The primary network link between the two sites has failed.
3. **Alternate link from A to B is down** — The backup/alternate route between sites is also unavailable.
4. **The message has been lost** — Both sites and all links are up, but messages are being dropped in transit.

2 Steps of Failure Reconfiguration

2 Steps of Failure Reconfiguration
1. **Step 1 — Broadcast link failure** — If the link from A to B has failed, this must be broadcast to every site in the system.
2. **Step 2 — Broadcast site failure** — If a site has failed, every other site must be notified that services offered by the failed site are no longer available. When recovered, broadcast restoration.

3 Distributed Systems Design Goals

3 Distributed Systems Design Goals
1. **Transparency** — System appears as a conventional centralized system to users — distribution is hidden.
2. **Fault Tolerance** — System continues to function in the face of partial failures of sites or links.
3. **Scalability** — As demands increase, system easily accepts addition of new resources (e.g., Hadoop framework).

DFS Challenges and Remote Access Details

DFS Challenges and Remote Access Details
1. Naming and Transparency — Location transparency (name does not reveal location) and location independence (name does not change when location changes). Multilevel mapping hides storage details.
2. Remote File Access — Caching — Recently accessed disk blocks retained locally. If not cached: copy from server. Accesses on cached copy. One master copy on server; copies scattered in client caches.
3. Cache-Consistency Problem — Keeping cached copies consistent with master. Can be called "network virtual memory." Client-initiated or server-initiated approaches.