

Unit 4: Computer Systems and Networks

Unit 4 explains how computers connect and cooperate. It covers the Internet as a network of autonomous systems, how packets carry data across it, how IP addresses and DNS find destinations, the protocols that standardize communication, how the network survives failures, how parallel and distributed computing speed up hard problems, and how systems scale to meet demand.

AP Computer Science Principles

Computer Systems and Networks

About 11 minutes to read

How to use this guide

Read it in order the first time because the ideas build on each other. Packets explain how data moves, IP addresses and DNS explain how it is addressed, protocols explain how machines agree on the rules, fault tolerance explains why the whole thing survives damage, and parallel computing explains how big problems get split up.

After the first read, use the trap boxes and the comparison tables to review the distinctions the exam tests most often, like TCP versus UDP or parallel versus distributed computing. Finish with the practice questions, then complete the recall check on the last page out loud and note any items you cannot explain yet.

What this unit is worth. Computer Systems and Networks is 11 to 15 percent of the AP Computer Science Principles exam. That makes it a mid-sized unit, but its vocabulary shows up everywhere. Questions about data, programming, and the impact of computing borrow words like protocol, packet, and redundancy, so a shaky definition here can cost points in units you thought you knew.

4.1 The Internet Is a Network of Autonomous Systems

The Internet is not one network owned by one company. It is a network of networks. Each participating network is run by its own organization, an Internet service provider, a university, a company, or a government agency. Each independently managed network is called an **autonomous system**. An autonomous system runs its own routers and makes its own routing decisions, and it connects to other autonomous systems so that data can travel between them.

No single authority controls the Internet end to end. The systems cooperate because they all follow the same **protocols**, agreed-upon standards for communication, so data can cross organizational boundaries without anyone being in charge of the whole trip. Routing decisions are made locally by each autonomous system, and packets find their way across many independently run networks on the way from sender to receiver. This decentralized design is also what makes the Internet fault tolerant, as you will see in the section on redundancy.

Trap. The Internet has no central owner and no master router. If a question asks who controls the Internet, the correct answer will never name a single company or government. Decentralization is the point, and it is what lets the network survive the failure of any one part.

Packets: How Data Travels

When you send data across the Internet, it does not travel as one continuous stream. The sending device breaks the data into small chunks called **packets**. Each packet carries three things: the chunk of data being transported, the source and destination IP addresses, and sequence information that records where its chunk belongs in the original message. That sequence information is what lets the receiving device reassemble the chunks in the right order.

Routers move packets from network to network. A router reads a packet's destination address and forwards it to the next router on the path, choosing among its connections. Packets from the same message do not have to take the same path. They can split across different routes, and because some routes are slower or more congested than others, packets can arrive out of order. The receiver uses the sequence information to put them back together, and if a packet never arrives, the receiver can ask for it to be sent again.

Trap. There is no dedicated path set up between sender and receiver. Each packet is routed independently, possibly over a different path, and the receiver reassembles them. If an answer choice assumes one fixed route for the whole message, it is wrong.

IP Addresses

Every device that communicates on the Internet needs an address, and that address is the **IP address**. Routers read the destination IP address on each packet to decide where to send it next. Without IP addresses, a router would have no way to aim a packet at its destination.

IP addresses are assigned to devices, often dynamically. When a device joins a network, it is typically given an IP address for the duration of its connection, and it may receive a different one the next time it connects. An IP address identifies where a device is attached to the network right now, not the device itself permanently.

Trap. An IP address is not a permanent serial number for a device. It is closer to a hotel room number. It tells the network where to deliver packets right now, and it can change the next time the device connects.

DNS: Turning Names into Addresses

People remember names like `rycal.web.app`, but routers only understand IP addresses. The **Domain Name System (DNS)** is the distributed service that translates domain names into IP addresses. When you type a web address, your device asks DNS for the matching IP address before any page data is sent.

DNS is **hierarchical** and **distributed**. No single server holds every name. At the top of the hierarchy are root servers, which direct queries to servers for top-level domains like `.com` and `.org`, which in turn direct queries to the authoritative servers for individual domains. A lookup walks down this hierarchy until it reaches a server that knows the answer. Because the work is spread across many servers, no single failure takes down the whole system, and because answers are cached along the way, repeat visits to the same site resolve faster.

Trap. DNS translates names into addresses. It does not route packets, encrypt data, or verify identity. On the exam, a question about finding the IP address for a domain name points to DNS. A question about moving packets toward that address points to routing.

Protocols: The Rules Every Device Follows

A **protocol** is an agreed-upon set of rules for communication. Protocols are what let devices built by different companies, running different software, exchange data at all. Without them, autonomous systems could not cooperate, because nothing would agree on how a packet should be formatted or what an acknowledgment means.

TCP: Reliable, Ordered Delivery

The **Transmission Control Protocol (TCP)** is used when every byte has to arrive, and arrive in order. It provides reliability through **acknowledgments** and **retransmission**. The receiver confirms the packets it gets, and if the sender does not receive an acknowledgment in time, it sends the packet again. The receiver also uses sequence numbers to reorder packets that arrived out of order before passing the data along. File downloads, email, and web pages use TCP, because a missing or scrambled chunk would corrupt the result.

UDP: Fast, Best-Effort Delivery

The **User Datagram Protocol (UDP)** skips the reliability machinery. It sends packets without acknowledgments, retransmission, or ordering. That makes it faster and lower-overhead, which is what real-time applications need. A video call or an online game would rather drop a frame than freeze while waiting for a retransmission, so a lost packet is simply skipped. DNS lookups also usually travel over UDP, because a single small request and response does not justify TCP's setup cost.

PROTOCOL	WHAT IT GUARANTEES	COST	TYPICAL USE
TCP	Every packet arrives, in order, through acknowledgments and retransmission	Slower, more overhead	Web pages, email, file transfers
UDP	None. Packets may be lost, duplicated, or out of order	Faster, less overhead	Video calls, online games, DNS lookups

HTTP and HTTPS

The **Hypertext Transfer Protocol (HTTP)** is the protocol browsers and servers use to request and deliver web pages. **HTTPS** is HTTP with encryption added, so the data cannot be read or modified in transit. The S stands for Secure, and the encryption is what keeps a password or credit card number private between your browser and the site. A login page served over plain HTTP sends credentials in readable form to anyone watching the network.

Trap. TCP and UDP are not better and worse. They are different tradeoffs. When a question asks which protocol fits a scenario, match the need: perfect accuracy and order means TCP, speed with tolerance for loss means UDP. The most common wrong answer picks TCP for a video call or UDP for a file transfer.

4.2 Fault Tolerance and Redundancy

A system is **fault tolerant** if it keeps working when parts of it fail. The Internet achieves fault tolerance through **redundancy**, which means having more than one way to get the job done. There are many paths between any two points on the Internet, so if one router fails or one link goes down, packets are simply routed around the damage along a different path. The failure may slow things down or force a detour, but the network as a whole keeps delivering data.

This is where the decentralized design pays off. A network with a single central router would stop entirely when that router failed. The Internet's many autonomous systems, each with multiple connections to others, mean there is no single point whose failure kills the network. Redundancy is the same idea in smaller systems: a server with two power supplies, or a file stored in two places, survives a failure that would stop a non-redundant setup.

FAILURE	HOW REDUNDANCY HANDLES IT
A router fails	Routers forward packets along alternate paths around it
A link becomes congested	Routers choose less busy routes for new packets
A server goes down	Requests go to a mirrored server holding the same data

Trap. Fault tolerance does not mean nothing ever fails or slows down. It means the system keeps functioning despite failures. If a question describes a router failing and asks what happens, the answer is that traffic reroutes around it, not that the network stops or that the data is lost for good.

4.3 Sequential, Parallel, and Distributed Computing

Sequential computing does one operation at a time, in order. A single processor working through a list of instructions is sequential. **Parallel computing** splits a task across multiple processors working at the same time, usually several cores inside one machine. **Distributed computing** splits a task across multiple machines connected by a network, each doing part of the work.

The payoff of parallel computing is **speedup**, which means the task finishes faster than it would sequentially. If a job takes 100 seconds on one processor and 25 seconds on four, the speedup is a factor of 4. But speedup is almost never perfectly proportional to the number of processors. Three things eat into it. First, **overhead**: splitting the work up and combining the results takes time. Second, **communication**: processors have to coordinate and share data, which costs more as more processors join. Third, some parts of a task simply cannot be parallelized and must still run sequentially, which caps how fast the whole thing can go no matter how many processors you add.

MODEL	WHERE THE WORK HAPPENS	EXAMPLE
Sequential	One processor, one step at a time	Sorting a list on a single core
Parallel	Multiple processors in one machine	Rendering video frames across 8 cores
Distributed	Multiple machines over a network	Thousands of computers jointly analyzing telescope data

Trap. Twice as many processors does not mean twice as fast. Because of overhead, communication costs, and the sequential portion of the task, doubling processors gives less than double the speedup. If an answer choice claims perfectly linear speedup, cross it out.

Scalability

Scalability is the ability of a system to expand or contract to meet demand. A scalable system handles growth gracefully: when more users arrive or more data needs processing, the system adds capacity and keeps performing. Cloud services are the classic example. A website on scalable infrastructure rents more servers during a traffic spike and releases them when the spike passes, instead of collapsing under the load.

Scalability connects to the rest of the unit. Distributed computing is one way to scale, because adding machines adds capacity. Redundancy helps a system stay up while it scales. When the exam asks whether a system is scalable, it is asking whether the system can grow without being redesigned.

Trap. Scalability is about handling growth, not about speed on a fixed task. A fast program that breaks when the input doubles is fast but not scalable. Keep handling more separate from running faster.

Confusions That Cost Points

PAIR	HOW TO KEEP THEM STRAIGHT
TCP vs UDP	TCP guarantees every packet arrives in order through acknowledgments and retransmission. UDP sends without guarantees and is faster. Match the protocol to the need: accuracy means TCP, real-time speed means UDP.
DNS vs routing	DNS translates a domain name into an IP address. Routing moves packets toward that address. Name lookup versus packet delivery.
IP address vs domain name	The domain name is the human-readable label. The IP address is the numeric destination routers use. DNS converts the first into the second.
Fault tolerance vs redundancy	Redundancy is the means: extra paths, extra copies. Fault tolerance is the result: the system keeps working when something fails.
Parallel vs distributed	Parallel splits work across processors in one machine. Distributed splits it across machines on a network. Neither gives perfectly linear speedup.

Practice Questions

Original questions written for this guide in the style of the AP exam. Answers and explanations are on the next page, so complete the questions before checking them.

1. A student is building a live video-chat feature for a study app. Occasional dropped frames are acceptable, but delays are not. Which protocol should the app use for the video stream, and why?

1. UDP, because it avoids the delay caused by acknowledgments and retransmission
2. TCP, because it guarantees that every frame arrives in order
3. TCP, because it is faster than UDP for real-time data
4. UDP, because it encrypts the video stream

2. A file is sent from one computer to another across the Internet. Which of the following best describes how the file's data travels?

1. The file travels along a single dedicated path reserved for the transfer
2. The file is broken into packets that may take different paths and are reassembled at the destination
3. The file is sent as one continuous stream that routers forward without dividing it
4. The file is copied to every router along the way, and the destination keeps the fastest copy

3. When a user types `www.example.com` into a browser, what is the role of DNS in loading the page?

1. DNS encrypts the connection between the browser and the server
2. DNS breaks the page into packets and routes them to the user
3. DNS translates the domain name into the IP address of the server
4. DNS gives the user's device a new IP address each time it connects to the network

4. A router in the middle of the Internet fails unexpectedly. Which of the following is the most likely outcome?

1. All data traveling through that router is permanently lost, and the transfers fail
2. The entire Internet stops working until the router is repaired
3. Senders must wait for the router to be repaired before sending any new data
4. Packets are rerouted around the failed router along other paths, and delivery continues

5. A program takes 60 seconds to run on one processor. When run on 4 processors, it takes 20 seconds instead of the 15 seconds that perfectly proportional speedup would predict. Which of the following best explains the difference?

1. The processors run more slowly when they work together
2. Overhead from splitting the work, communication between processors, and portions of the task that cannot be parallelized
3. The program contains a syntax error that only appears in parallel mode
4. Distributed computing is always slower than sequential computing

6. A scientist has a huge set of telescope images to analyze. She can either rent time on a server with 32 processor cores or split the task across thousands of home computers connected over the Internet. Which statement best compares the two approaches?

1. The server is distributed computing and the home computers are parallel computing
2. Both approaches are parallel computing because both use more than one processor
3. The server is parallel computing and the home computers are distributed computing, and neither one guarantees perfectly proportional speedup
4. The home computers must finish faster because the Internet moves data instantly

7. An online store expects ten times its usual traffic during a holiday sale. Which choice best shows the store's system is scalable?

1. It runs on cloud infrastructure that adds servers as traffic grows and releases them when traffic falls
2. It runs on the single fastest server the company could buy
3. It stores its product images on the fastest available hard drive
4. It splits each customer's checkout across multiple processors so orders finish faster

8. Which statement best describes how data travels from a sender to a receiver across the Internet?

1. A central authority plans the best route for the data before anything is sent
2. The sender's Internet service provider controls every router the data passes through
3. DNS servers carry the data from one router to the next until it arrives
4. Each autonomous system makes its own routing decisions, and routers forward each packet independently toward its destination

Answer Key

- 1. A.** Video chat needs speed more than perfect accuracy, and UDP skips acknowledgments and retransmission, so it avoids the delays that would freeze the stream. B picks the protocol whose reliability machinery causes exactly the delay the scenario rules out. C reverses the speed tradeoff: TCP's reliability makes it slower, not faster. D confuses layers: UDP does not encrypt anything, and encryption is not what the scenario is asking about.
- 2. B.** Data is broken into packets at the source, each packet is routed independently and may take a different path, and the destination reassembles them using sequence information. A describes circuit switching, which is not how the Internet works. C is wrong because the data is divided into packets rather than sent as one stream. D invents a mechanism that does not exist; routers forward packets, they do not stockpile copies for the destination to choose from.
- 3. C.** DNS translates the human-readable domain name into the IP address routers need. A confuses DNS with HTTPS, which provides the encryption. B confuses DNS with routing and packetization, which happen after the address is known. D confuses DNS with how devices receive IP addresses: an address is assigned when a device joins a network, and handing out addresses is not DNS's job.
- 4. D.** Redundancy means many paths exist between any two points, so routers forward packets around the failed router and delivery continues. A ignores both rerouting and TCP retransmission, which recovers packets lost in the failure. B contradicts the decentralized design: there is no single point whose failure stops the whole Internet. C is wrong because new data routes around the failure too; nothing has to wait.
- 5. B.** Speedup is sublinear because splitting work and recombining results costs time, processors spend time communicating and coordinating, and the sequential portion of the task cannot be sped up at all. A is wrong because processors do not slow down; coordination overhead is the cost. C is wrong because a syntax error would prevent the program from running, not merely slow it. D is wrong on two counts: this scenario is parallel, not distributed, computing, and neither is always slower than sequential.
- 6. C.** Thirty-two cores inside one machine is parallel computing, and thousands of machines connected over the Internet is distributed computing. Neither approach guarantees perfectly proportional speedup, because splitting the work and recombining results costs time, processors spend time coordinating, and some of the task must still run sequentially. A swaps the two terms. B is wrong because distributed computing means multiple machines over a network, not processors in one machine. D is wrong because sending data over a network adds communication cost; the Internet does not move data instantly.
- 7. A.** Scalability is the ability to handle growth without being redesigned. Cloud infrastructure that adds servers during the spike and releases them afterward grows with demand. B buys speed, not scalability: one fast server still breaks when traffic exceeds what it can handle. C speeds up reads from storage but does not add capacity for more users. D describes parallel computing speeding up a fixed task, which is a different idea from handling growth.
- 8. D.** The Internet is a network of independently run autonomous systems with no central authority. Each system makes its own routing decisions, and routers forward each packet independently, possibly along different paths, toward the destination. A is wrong because no central authority plans routes. B is wrong because an Internet service provider controls only its own autonomous system, not routers on other networks. C is wrong because DNS translates names into IP addresses; routers, not DNS servers, move packets.

When you check your answers, note which distinction each miss came from. Make a flashcard for that distinction and drill it spaced out over the next few days instead of rereading the whole section. If you missed one of these questions, the same distinction is worth practicing again in Rycal, where the Computer Systems and Networks deck has flashcards for it and more practice questions use the same kinds of traps.

One-Page Recall Check

Say each answer out loud before you look back, and mark the ones you cannot finish. Anything you cannot say out loud yet belongs in your flashcard deck. In Rycal, add those items to the Computer Systems and Networks deck and let spaced review bring them back over the next few days.

- Explain what an autonomous system is and why the Internet has no central owner.
- Name the three things a packet carries and explain what each one is for.
- Explain why packets from one message can arrive out of order and how the receiver handles it.
- Explain how IP addresses are assigned and why one can change over time.
- Describe the hierarchy DNS uses to turn a domain name into an IP address.
- State what a protocol is and why protocols are necessary on the Internet.
- Explain how TCP achieves reliable, ordered delivery.
- Explain why a video call uses UDP instead of TCP.
- Explain what HTTPS adds to HTTP.
- Define fault tolerance and explain the role of redundancy.
- Describe what happens to packets when a router fails.
- Distinguish sequential, parallel, and distributed computing, with an example of each.
- Explain why adding processors does not give perfectly linear speedup, naming all three causes.
- Define scalability and give an example.

Where to go next. Turn every missed item above into flashcards and drill them spaced out over several days rather than in one sitting. In Rycal, open the Computer Systems and Networks deck under AP Computer Science Principles. The deck covers the terms in this guide, and its practice questions target the same traps named here. If you have a test date, add it in the Test Planner. You can also start your next review with a Brain Dump, then check what you missed against this guide.

Key terms for this unit

Autonomous system, Packet, Router, IP address, Domain Name System (DNS), Protocol, Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Hypertext Transfer Protocol (HTTP), HTTPS, Acknowledgment, Retransmission, Fault tolerance, Redundancy, Sequential computing, Parallel computing, Distributed computing, Speedup, Scalability.

About this guide. Written for Rycal and aligned to the College Board AP Computer Science Principles course framework, Unit 4. All questions and explanations are original Rycal writing. Rycal is independent and is not affiliated with or endorsed by the College Board.