UCLA-ALIGNED COURSE FIELD NOTES

CS 118

Computer Network Fundamentals

Layering, applications, transport, IP, routing, congestion, Ethernet, wireless, performance, and security

How these notes are written. I wrote each chapter the way I wish the material had been explained the first time: the real idea, the point where I usually get stuck, the simplest mental model that still stays honest, and a worked decision instead of a polished answer appearing from nowhere. Every chapter closes with practice and a fully written solution. If a sentence sounds impressive but does not help me solve or explain something, it does not belong here.

Daniel Mastick

Curriculum map, working notes, practice, and solutions

Course map

Week	Core thread	What should be solid by the end
1	**Network architecture, layering, and performance**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
2	**Application protocols, DNS, and HTTP**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
3	**Sockets, UDP, and reliable protocol design**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
4	**TCP reliability, flow control, and connection state**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
5	**Congestion control and queue management**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
6	**Internet protocol, addressing, fragmentation, and NAT**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
7	**Routing algorithms and Internet routing**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
8	**Link layers, Ethernet, switching, and wireless**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
9	**Measurement, security, and network debugging**	Reconstruct the model, work an application, test its assumptions, and explain the result in ordinary language.
9	**Integration studio**	Combine several chapters in one case, preserve their assumptions, and reconcile the result across representations.
10	**Cumulative review**	Retrieve the full course map from memory and solve mixed problems without chapter labels.

Scope anchor: https://catalog.registrar.ucla.edu/course/2025/COMSCI118?year=2025. The sequence is aligned to the official catalog description and expanded into a practical ten-week study plan.

How I would use this packet

Treat each numbered section as one chapter. Read the formal explanation, pause at the student interpretation, and see whether the easy version still says the same thing. Rework the example without looking, then cover the solution in the chapter practice box. The cumulative set at the end is deliberately mixed: knowledge becomes durable when I have to choose the tool instead of being told which chapter I am in.

Contents

CHAPTER 1 Network architecture, layering, and performance

The idea

Layering separates service from implementation through protocols and interfaces. Encapsulation, end-to-end design, packet switching, delay, loss, throughput, the bandwidth and delay product, and statistical multiplexing define the network's basic tradeoffs.

Rebuild the chapter from first principles

The claim I need to own is this: Layering separates service from implementation through protocols and interfaces. Encapsulation, end-to-end design, packet switching, delay, loss, throughput, the bandwidth and delay product, and statistical multiplexing define the network's basic tradeoffs. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of network architecture, layering, and performance. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach network architecture, layering, and performance on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from the course foundations to application protocols, dns, and http. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Layering separates service from implementation through protocols and interfaces.
- **What moves.** Encapsulation, end-to-end design, packet switching, delay, loss, throughput, the bandwidth and delay product, and statistical multiplexing define the network's basic tradeoffs.
- **What keeps the answer honest.** The model becomes useful only when I can apply network architecture, layering, and performance to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I trace one packet down, across, and back up the stack; that keeps every header and responsibility in the right layer.

The easy version

Layering separates service from implementation through protocols and interfaces.

Worked example

Question. A 1500-byte packet crosses a 12 Mbps link. Find serialization delay.
Walkthrough. The packet has 12000 bits. Transmission delay is 12000 divided by 12 million, or 1 millisecond, excluding propagation, queueing, and processing.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. A 1500-byte packet crosses a 12 Mbps link. Find serialization delay.
Solution. The packet has 12000 bits. Transmission delay is 12000 divided by 12 million, or 1 millisecond, excluding propagation, queueing, and processing.

Practice 2. Explain network architecture, layering, and performance to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct network architecture, layering, and performance from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a network architecture, layering, and performance problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 2 Application protocols, DNS, and HTTP

The idea

Network applications define messages, syntax, semantics, state, naming, caching, and failure. DNS resolves names through hierarchical distributed caching; HTTP combines methods, status, headers, representations, connections, proxies, and security through TLS.

Rebuild the chapter from first principles

The claim I need to own is this: Network applications define messages, syntax, semantics, state, naming, caching, and failure. DNS resolves names through hierarchical distributed caching; HTTP combines methods, status, headers, representations, connections, proxies, and security through TLS. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of application protocols, dns, and http. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach application protocols, dns, and http on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from network architecture, layering, and performance to sockets, udp, and reliable protocol design. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Network applications define messages, syntax, semantics, state, naming, caching, and failure.
- **What moves.** DNS resolves names through hierarchical distributed caching; HTTP combines methods, status, headers, representations, connections, proxies, and security through TLS.

- **What keeps the answer honest.** The model becomes useful only when I can apply application protocols, dns, and http to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I distinguish application state from transport connections; one user session can survive many TCP connections.

The easy version

Network applications define messages, syntax, semantics, state, naming, caching, and failure.

Worked example

Question. Why can a DNS answer remain after the authoritative record changes?
Walkthrough. Resolvers cache records until time-to-live expiry. The distributed cache trades freshness for lower latency and server load.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why can a DNS answer remain after the authoritative record changes?
Solution. Resolvers cache records until time-to-live expiry. The distributed cache trades freshness for lower latency and server load.

Practice 2. Explain application protocols, dns, and http to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct application protocols, dns, and http from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a application protocols, dns, and http problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent

check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 3 Sockets, UDP, and reliable protocol design

The idea

Sockets expose endpoints to applications. UDP provides datagrams with checksums and multiplexing but no ordering, retransmission, flow control, or congestion control; applications add only the reliability semantics they need.

Rebuild the chapter from first principles

The claim I need to own is this: Sockets expose endpoints to applications. UDP provides datagrams with checksums and multiplexing but no ordering, retransmission, flow control, or congestion control; applications add only the reliability semantics they need. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of sockets, udp, and reliable protocol design. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.

- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach sockets, udp, and reliable protocol design on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from application protocols, dns, and http to tcp reliability, flow control, and connection state. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Sockets expose endpoints to applications.
- **What moves.** UDP provides datagrams with checksums and multiplexing but no ordering, retransmission, flow control, or congestion control; applications add only the reliability semantics they need.
- **What keeps the answer honest.** The model becomes useful only when I can apply sockets, udp, and reliable protocol design to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: unreliable does not mean useless; it means the application owns the missing guarantees.

The easy version

Sockets expose endpoints to applications.

Worked example

Question. A real-time voice packet arrives late. Why might dropping beat retransmission?
Walkthrough. Playback has a deadline. A retransmitted old packet may arrive after it can be used, while waiting adds latency and jitter to current audio.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. A real-time voice packet arrives late. Why might dropping beat retransmission?
Solution. Playback has a deadline. A retransmitted old packet may arrive after it can be used, while waiting adds latency and jitter to current audio.

Practice 2. Explain sockets, udp, and reliable protocol design to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct sockets, udp, and reliable protocol design from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a sockets, udp, and reliable protocol design problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 4 TCP reliability, flow control, and connection state

The idea

TCP provides a byte stream using sequence numbers, acknowledgments, retransmission timers, cumulative ACKs, windows, and connection state. Flow control protects the receiver; connection setup and teardown synchronize sequence spaces and state.

Rebuild the chapter from first principles

The claim I need to own is this: TCP provides a byte stream using sequence numbers, acknowledgments, retransmission timers, cumulative ACKs, windows, and connection state. Flow control protects the receiver; connection setup and teardown synchronize sequence spaces and state. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of tcp reliability, flow control, and connection state. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach tcp reliability, flow control, and connection state on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from sockets, udp, and reliable protocol design to congestion control and queue management. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** TCP provides a byte stream using sequence numbers, acknowledgments, retransmission timers, cumulative ACKs, windows, and connection state.
- **What moves.** Flow control protects the receiver; connection setup and teardown synchronize sequence spaces and state.

- **What keeps the answer honest.** The model becomes useful only when I can apply tcp reliability, flow control, and connection state to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I annotate byte ranges rather than packet numbers because TCP acknowledges bytes.

The easy version

TCP provides a byte stream using sequence numbers, acknowledgments, retransmission timers, cumulative ACKs, windows, and connection state.

Worked example

Question. Sender transmits bytes 1000 through 1499 and receiver sends ACK 1500. What does it mean?
Walkthrough. All bytes through 1499 arrived in order and the next expected byte is 1500. It does not necessarily reveal whether later out-of-order bytes are buffered.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Sender transmits bytes 1000 through 1499 and receiver sends ACK 1500. What does it mean?
Solution. All bytes through 1499 arrived in order and the next expected byte is 1500. It does not necessarily reveal whether later out-of-order bytes are buffered.

Practice 2. Explain tcp reliability, flow control, and connection state to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct tcp reliability, flow control, and connection state from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a tcp reliability, flow control, and connection state problem but never states a model or checks the result. Write the shortest

useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 5 Congestion control and queue management

The idea

Congestion control protects the network through slow start, congestion avoidance, loss or delay signals, multiplicative decrease, pacing, and fairness goals. Receiver flow control and network congestion control solve different bottlenecks.

Rebuild the chapter from first principles

The claim I need to own is this: Congestion control protects the network through slow start, congestion avoidance, loss or delay signals, multiplicative decrease, pacing, and fairness goals. Receiver flow control and network congestion control solve different bottlenecks. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of congestion control and queue management. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical

example. If one representation disagrees with another, that disagreement is useful evidence.

- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach congestion control and queue management on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from tcp reliability, flow control, and connection state to internet protocol, addressing, fragmentation, and nat. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Congestion control protects the network through slow start, congestion avoidance, loss or delay signals, multiplicative decrease, pacing, and fairness goals.
- **What moves.** Receiver flow control and network congestion control solve different bottlenecks.
- **What keeps the answer honest.** The model becomes useful only when I can apply congestion control and queue management to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: the receiver window says what the endpoint can absorb; the congestion window says what the path can absorb.

The easy version

Congestion control protects the network through slow start, congestion avoidance, loss or delay signals, multiplicative decrease, pacing, and fairness goals.

Worked example

Question. Why does TCP increase cautiously after reaching steady operation?
Walkthrough. Aggressive growth would overflow shared queues and destabilize competing flows. Additive increase probes capacity while multiplicative decrease reacts strongly to congestion.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why does TCP increase cautiously after reaching steady operation?
Solution. Aggressive growth would overflow shared queues and destabilize competing flows. Additive increase probes capacity while multiplicative decrease reacts strongly to congestion.

Practice 2. Explain congestion control and queue management to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct congestion control and queue management from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a congestion control and queue management problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 6 Internet protocol, addressing, fragmentation, and NAT

The idea

IP offers best-effort datagrams across heterogeneous links. Prefix addressing, subnetting, longest-prefix match, DHCP, ARP or neighbor discovery, TTL, ICMP, MTU, fragmentation, IPv6, and NAT connect local delivery to internetworking.

Rebuild the chapter from first principles

The claim I need to own is this: IP offers best-effort datagrams across heterogeneous links. Prefix addressing, subnetting, longest-prefix match, DHCP, ARP or neighbor discovery, TTL, ICMP, MTU, fragmentation, IPv6, and NAT connect local delivery to internetworking. I do not count that as learned

just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of internet protocol, addressing, fragmentation, and nat. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach internet protocol, addressing, fragmentation, and nat on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from congestion control and queue management to routing algorithms and internet routing. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** IP offers best-effort datagrams across heterogeneous links.
- **What moves.** Prefix addressing, subnetting, longest-prefix match, DHCP, ARP or neighbor discovery, TTL, ICMP, MTU, fragmentation, IPv6, and NAT connect local delivery to internetworking.
- **What keeps the answer honest.** The model becomes useful only when I can apply internet protocol, addressing, fragmentation, and nat to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I split address, prefix, next hop, and link-layer destination; they are four different questions.

The easy version

IP offers best-effort datagrams across heterogeneous links.

Worked example

Question. How many usable addresses does an IPv4 /24 conventionally provide?
Walkthrough. A /24 leaves eight host bits, giving 256 addresses. Traditional subnet and broadcast reservations leave 254 usable host addresses, though point-to-point and modern practices vary.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. How many usable addresses does an IPv4 /24 conventionally provide?
Solution. A /24 leaves eight host bits, giving 256 addresses. Traditional subnet and broadcast reservations leave 254 usable host addresses, though point-to-point and modern practices vary.

Practice 2. Explain internet protocol, addressing, fragmentation, and nat to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct internet protocol, addressing, fragmentation, and nat from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and

choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a internet protocol, addressing, fragmentation, and nat problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 7 Routing algorithms and Internet routing

The idea

Distance-vector and link-state algorithms compute paths under different information models. Intra-domain protocols optimize operator metrics; BGP exchanges policy-constrained reachability among autonomous systems, where business policy can outrank shortest path.

Rebuild the chapter from first principles

The claim I need to own is this: Distance-vector and link-state algorithms compute paths under different information models. Intra-domain protocols optimize operator metrics; BGP exchanges policy-constrained reachability among autonomous systems, where business policy can outrank shortest path. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of routing algorithms and internet routing. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach routing algorithms and internet routing on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from internet protocol, addressing, fragmentation, and nat to link layers, ethernet, switching, and wireless. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Distance-vector and link-state algorithms compute paths under different information models.
- **What moves.** Intra-domain protocols optimize operator metrics; BGP exchanges policy-constrained reachability among autonomous systems, where business policy can outrank shortest path.
- **What keeps the answer honest.** The model becomes useful only when I can apply routing algorithms and internet routing to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: internet routing is policy plus reachability, not one global Dijkstra run.

The easy version

Distance-vector and link-state algorithms compute paths under different information models.

Worked example

Question. Why can distance-vector suffer count-to-infinity?
Walkthrough. Routers with incomplete local views can repeatedly believe a failed destination remains reachable through one another, increasing metrics step by step. Split horizon and poison reverse reduce some loops.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. Why can distance-vector suffer count-to-infinity?
Solution. Routers with incomplete local views can repeatedly believe a failed destination remains reachable through one another, increasing metrics step by step. Split horizon and poison reverse reduce some loops.

Practice 2. Explain routing algorithms and internet routing to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct routing algorithms and internet routing from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a routing algorithms and internet routing problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 8 Link layers, Ethernet, switching, and wireless

The idea

Link protocols frame packets, detect errors, arbitrate shared media, and deliver across one hop. Ethernet learning switches build forwarding tables from source addresses; wireless adds interference, fading, hidden terminals, association, and rate adaptation.

Rebuild the chapter from first principles

The claim I need to own is this: Link protocols frame packets, detect errors, arbitrate shared media, and deliver across one hop. Ethernet learning switches build forwarding tables from source addresses; wireless adds interference, fading, hidden terminals, association, and rate adaptation. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of link layers, ethernet, switching, and wireless. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach link layers, ethernet, switching, and wireless on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from routing algorithms and internet routing to measurement, security, and network debugging. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Link protocols frame packets, detect errors, arbitrate shared media, and deliver across one hop.
- **What moves.** Ethernet learning switches build forwarding tables from source addresses; wireless adds interference, fading, hidden terminals, association, and rate adaptation.
- **What keeps the answer honest.** The model becomes useful only when I can apply link layers, ethernet, switching, and wireless to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

My shorthand for this chapter is: a switch learns from where frames came from, then forwards based on where they need to go.

The easy version

Link protocols frame packets, detect errors, arbitrate shared media, and deliver across one hop.

Worked example

Question. What does a learning switch do with an unknown destination MAC?
Walkthrough. It floods the frame on other ports in the broadcast domain while learning the source address and ingress port. A later reply helps populate the destination mapping.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. What does a learning switch do with an unknown destination MAC?
Solution. It floods the frame on other ports in the broadcast domain while learning the source address and ingress port. A later reply helps populate the destination mapping.

Practice 2. Explain link layers, ethernet, switching, and wireless to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct link layers, ethernet, switching, and wireless from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a link layers, ethernet, switching, and wireless problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 9 Measurement, security, and network debugging

The idea

Ping, traceroute, packet capture, logs, counters, and active or passive measurement expose different layers. Threats include spoofing, interception, denial of service, route attacks, and insecure endpoints; encryption does not replace authentication or availability engineering.

Rebuild the chapter from first principles

The claim I need to own is this: Ping, traceroute, packet capture, logs, counters, and active or passive measurement expose different layers. Threats include spoofing, interception, denial of service, route attacks, and insecure endpoints; encryption does not replace authentication or availability engineering. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of measurement, security, and network debugging. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach measurement, security, and network debugging on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from link layers, ethernet, switching, and wireless to the cumulative course model. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

The full working model

- **Core claim.** Ping, traceroute, packet capture, logs, counters, and active or passive measurement expose different layers.
- **What moves.** Threats include spoofing, interception, denial of service, route attacks, and insecure endpoints; encryption does not replace authentication or availability engineering.

- **What keeps the answer honest.** The model becomes useful only when I can apply measurement, security, and network debugging to an unfamiliar case and defend the assumptions.
- **Mastery standard.** I can define the objects, rebuild the rule, work a fresh case, compare alternatives, and diagnose a broken solution without relying on recognition.

How I actually think about it

I debug from the closest confirmed layer outward: link, address, route, transport, name, then application.

The easy version

Ping, traceroute, packet capture, logs, counters, and active or passive measurement expose different layers.

Worked example

Question. A host can ping an IP address but not open a site by hostname. What should be checked first?
Walkthrough. The path and IP stack work, so inspect DNS configuration and responses, then application and TLS details. Packet capture can confirm whether queries leave and answers return.

Common miss

The fastest way to get this chapter wrong is to use the visible procedure without naming its assumption, units, time period, reference group, or boundary. I put that condition beside the work and verify the result independently.

Solved chapter practice

Practice 1. A host can ping an IP address but not open a site by hostname. What should be checked first?
Solution. The path and IP stack work, so inspect DNS configuration and responses, then application and TLS details. Packet capture can confirm whether queries leave and answers return.

Practice 2. Explain measurement, security, and network debugging to a classmate using one definition, one concrete example, one assumption, and one failure mode.
Solution. Begin with the core claim above, use the worked case as the concrete example, state the condition highlighted by the warning, and choose a failure where that condition does not hold. A complete explanation connects all four pieces instead of listing them.

Mastery practice A. Reconstruct measurement, security, and network debugging from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a measurement, security,

and network debugging problem but never states a model or checks the result. Write the shortest useful review of that solution.

Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

Cumulative study plan

I use this packet in three passes. First I reconstruct each model and work its examples without looking. Second I mix chapters so the tool is no longer named for me. Third I explain a complete case aloud, including assumptions, units, uncertainty, failure modes, and what evidence would change my conclusion.