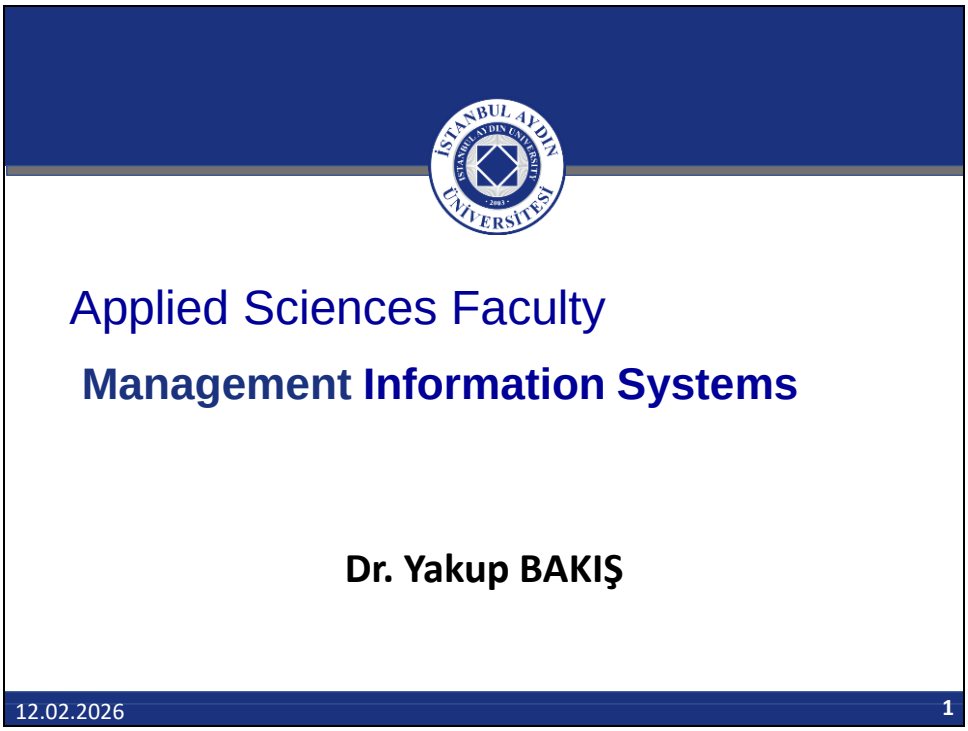


Good morning everyone.
Last week, we talked about how the Internet works in general.

Today, we move one layer higher.
We will begin to understand the **Application Layer**,
which is the layer closest to the user.

When you open a website, send an email, or use Instagram, you are using the Application Layer.

This is the part of networking that you actually see and use every day.

Chapter 1

Computer Networks and the Internet

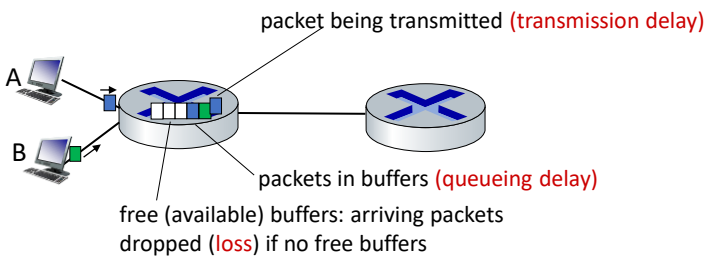
How many of you use the Internet every day?
Probably all of you.
But today, we will start learning how to build the web, not just use it.
That is the difference between a user and an engineer.
Users consume technology.
Engineers understand and create technology.

1.4 Delay, Loss, and Throughput in Packet-Switched Networks

How do packet loss and delay occur?

packets *queue* in router buffers

- packets queue, wait for turn
- arrival rate to link (temporarily) exceeds output link capacity: packet loss



Now we start a very important topic: **delay, loss, and throughput** in packet-switched networks.

Ideally, we would like data to move instantly with no loss.
But in reality, networks introduce delays and can drop packets

Look at this diagram.

In a packet-switched network, packets often arrive at a router faster than the router can send them out on the outgoing link. When this happens, packets must wait in a **buffer** (also called a **queue**) until it is their turn to be transmitted.

If packets arrive faster than the outgoing link can transmit them, they must wait.
This waiting time is called **queueing delay**. Queueing delay is not constant; it depends on how congested the router is at that moment. If many packets arrive close together, the queue grows and delay increases.

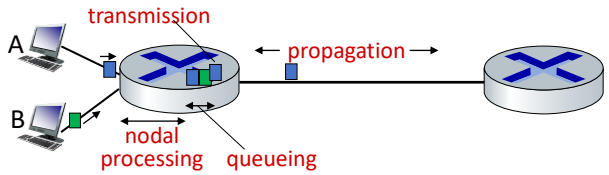
Buffers are finite. If the buffer becomes full and a new packet arrives, the router cannot store it, so the packet is **dropped**. This is **packet loss**.

Key idea: delay and loss are usually symptoms of **congestion**—too much

traffic for the available capacity.

1.4.1 Overview of Delay in Packet-Switched Networks

Packet delay: four sources



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc}: nodal processing
- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay
- time waiting at output link for transmission
- depends on congestion level of router

When a packet arrives at a router, it experiences several types of delay.

We call the total delay at a router the **nodal delay**.

There are four main sources:

1. Processing delay (dproc): The router examines the packet header, checks errors, and decides which output link to use. In high-speed routers this is typically microseconds or less.

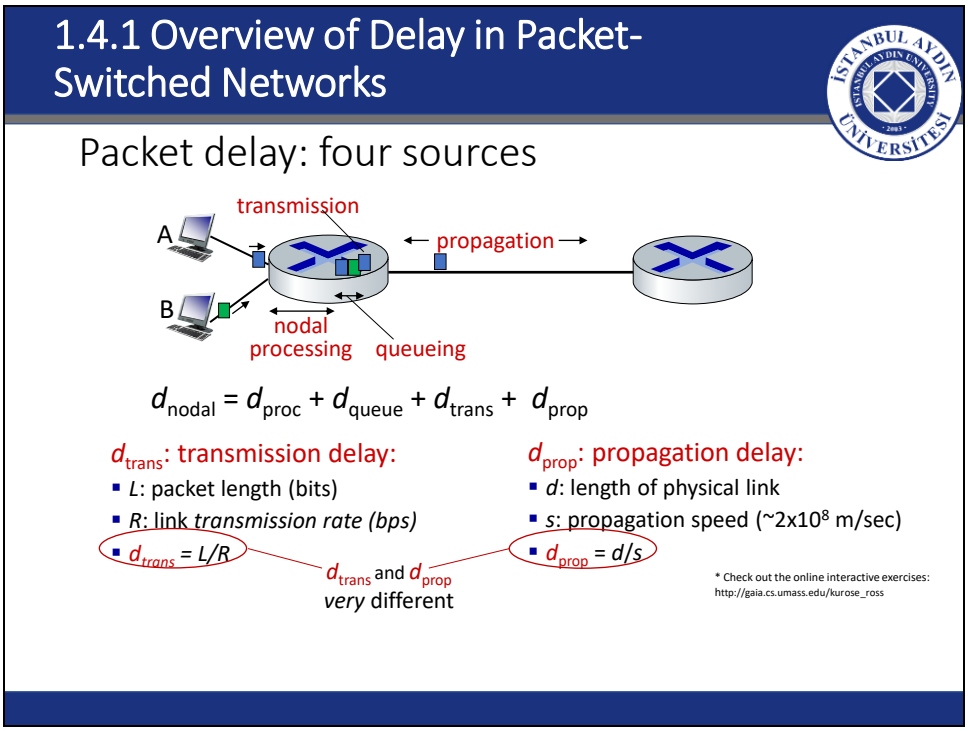
2.Queueing delay (dqueue): The time the packet waits in the queue before it can be transmitted. This depends heavily on congestion and can vary from packet to packet.

3.he time needed to push all packet bits onto the link. This depends on packet length and link rate: **dtrans = L / R**.

4.Propagation delay (dprop): The time for a bit to travel across the physical link. This depends on distance and propagation speed: **dprop = d / s**.

The total nodal delay is:
dnodal = dproc + dqueue + dtrans + dprop.

Different applications—like web browsing, maps, messaging, and VoIP—are strongly affected by these delays.



Now let's carefully compare two delays that Students often confuse **transmission delay** and **propagation delay**, but they are different.

Transmission delay (L/R): depends on how many bits are in the packet (**L**) and how fast the link can send bits (**R**). It does **not** depend on distance.
It depends on packet length **L** and link rate **R**:
dtrans = L / R.

Propagation delay (d/s): is the time for a bit to travel across the physical link.

how far the signal must travel (**d**) and the propagation speed (**s**), which is close to the speed of light in the medium.

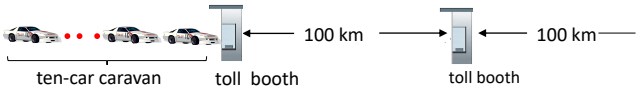
It does **not** depend on packet length.
It depends on distance **d** and propagation speed **s**:
dprop = d / s, where s is close to the speed of light in the medium.

Key point:
Transmission delay depends on **packet size and bandwidth**.
Propagation delay depends on **distance**.

Simple rule:
Bandwidth affects transmission delay; distance affects propagation delay.

1.4.2 Queuing Delay and Packet Loss

Caravan analogy



- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car ~ bit; caravan ~ packet
- *Q:* How long until caravan is lined up before 2nd toll booth?

- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr) = 1 hr
- *A:* 62 minutes

This slide uses a famous analogy to explain transmission and propagation.

This analogy makes the difference between transmission and propagation very clear. Think of:

•each car as a **bit**

•the whole caravan as a **packet**

•the toll booth as a **router**

The time to "push" the entire caravan through the toll booth is like transmission delay.

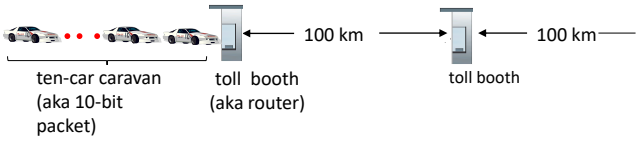
The time for cars to travel from one booth to the next is like propagation delay.

In this example, the total time is transmission delay plus propagation delay, giving **62 minutes**.

The important lesson is that total delay is often the sum of multiple different delays.

1.4.2 Queuing Delay and Packet Loss

Caravan analogy



- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- *Q:* Will cars arrive to 2nd booth before all cars serviced at first booth?
  A: Yes! after 7 min, first car arrives at second booth; three cars still at first booth

Now we change the parameters: cars travel much faster, and the toll booth is slower.

In this situation, the first car can arrive at the second booth
before the last cars even leave the first booth.

In networking terms, this means:
the first bits of a packet may arrive at the next router
while the remaining bits are still being transmitted.

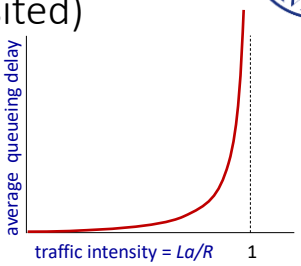
This helps you understand how transmission and propagation interact.

1.4.3 End-to-End Delay

Packet queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- *a:* average packet arrival rate

- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving is more than can be serviced - average delay infinite!



Now we focus on the most variable delay: **queueing delay**.

Queueing delay is the most variable part of nodal delay. To reason about it, the book defines the **traffic intensity**:

The traffic intensity is **La/R**.

L = packet length (bits)

a = average packet arrival rate (packets/sec)

R = link bandwidth (bits/sec)

If **La/R is close to 0**, packets rarely wait →the queue is usually empty, so delay is small.
If **La/R approaches 1**, the queue builds up often → queueing delay becomes very large.
If **La/R is greater than 1**, more work arrives than can be served, so the queue grows without bound—delay becomes effectively infinite.

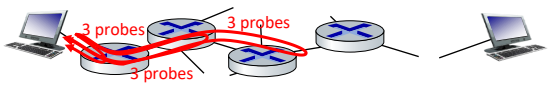
average delay becomes extremely large (conceptually "infinite").

Golden rule: Design systems so traffic intensity is not greater than 1.
keep the traffic intensity below 1 to avoid congestion collapse.

1.4.3 End-to-End Delay

"Real" Internet delays and routes

- what do "real" Internet delay & loss look like?
- **traceroute** program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i:*
  - sends three packets that will reach router *i* on path towards destination (with time-to-live field value of *i*)
  - router *i* will return packets to sender
  - sender measures time interval between transmission and reply



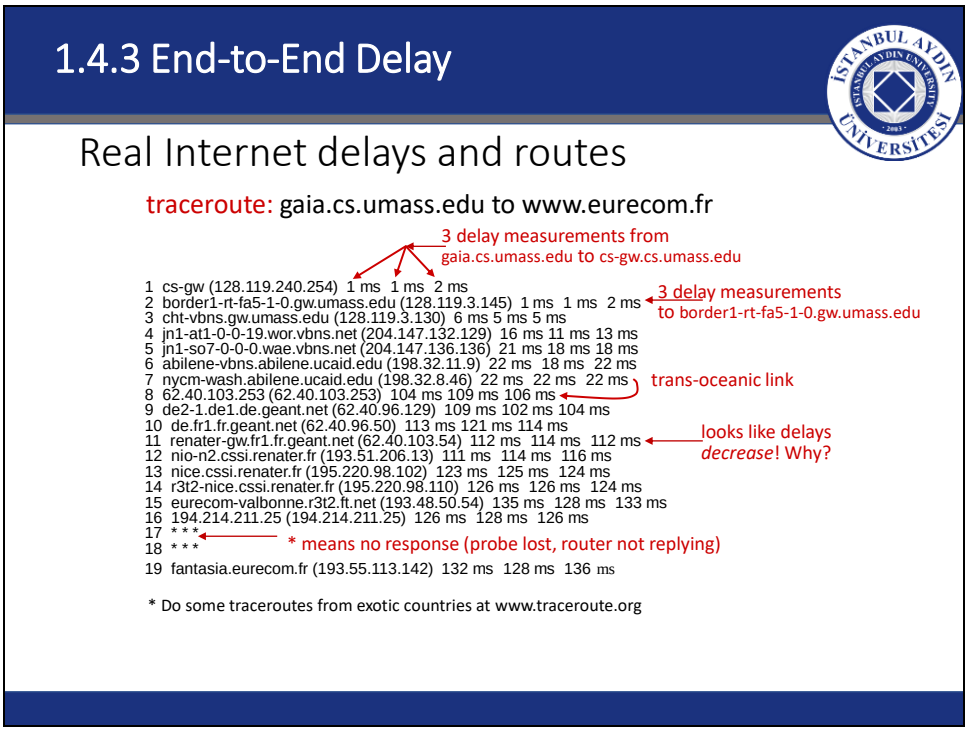
Now let's see how we can measure real Internet delay.
To see real Internet delay, we can use **traceroute**.

Traceroute measures the delay from a source host to each router along the path to a destination.

Traceroute sends packets with increasing **TTL (time-to-live)** values.
When a router receives such a packet, it sends a short reply back to the sender.

The sender measures the time between sending the probe and receiving the reply—this is the **round-trip time (RTT)**.

Traceroute typically sends **three probes per hop**, so you can see how the delay varies over time due to queueing and congestion.



This is an example output of traceroute.
Each line represents one hop (one router) along the path.

The last three columns show three RTT measurements.

RTT is the total time for a probe to go to a router and for the reply to come back.

They are often different because queueing delay changes from moment to moment.

An asterisk (*) means the probe did not receive a reply—either the probe was lost or that router does not respond.

Notice that sometimes a later hop can appear to have smaller delay than an earlier hop.

This does not mean the path got “shorter”; it happens because queueing delay is random and fluctuates.

Live Demo / Example Output Analysis

- C:\Users\ybaki>tracert google.com
- Tracing route to google.com [172.217.169.110]
- over a maximum of 30 hops:
- 1 <1 ms 1 ms <1 ms 192.168.0.1
- 2 8 ms 15 ms 11 ms 94.54.156.1
- 3 10 ms 11 ms 10 ms 172.25.44.81
- 4 9 ms 8 ms 11 ms 172.25.44.102
- 5 11 ms 9 ms 8 ms 172.25.99.25
- 6 8 ms 9 ms * 172.25.99.2
- 7 19 ms 17 ms 16 ms 46.197.15.154
- 8 40 ms 22 ms 20 ms 216.239.59.239
- 9 20 ms 18 ms 21 ms 216.239.49.199
- 10 17 ms 20 ms 17 ms sof02s31-in-f14.1e100.net [172.217.169.110]
- Trace complete.

What each line (hop) means

Each numbered line represents a **hop**.
A hop is typically a **router** (or a Layer-3 network device) that your packet reaches on the way to the destination.

•If the output ends at hop **10**, that means traceroute reached the destination in **10 hops**.

•If the output ends at hop **14**, that means traceroute reached the destination in **14 hops**.

Note: Hop count is not a perfect "distance" measure, but it is a useful indicator of how many networks/devices your traffic crosses.

12.02.2026 **Slide 11**

Traceroute / Tracert: How to Read and Interpret the Output (Detailed Guide)

Goal:

The purpose of traceroute (Windows: tracert) is to discover the path that packets follow from your computer to a destination host on the Internet, and to measure the delay to each hop along that path.

1) What you need to run

•On **Windows**, use:
tracert domain.com
Example: tracert google.com

•On **macOS/Linux**, use:
traceroute domain.com

Important: Do **not** include https:// or a full web page path.
Traceroute works with hostnames such as:

•google.com

•www.aydin.edu.tr

It will not work with:

•https://www.aydin.edu.tr/Pages/default.aspx

2) What each line (hop) means

Each numbered line represents a **hop**.
A hop is typically a **router** (or a Layer-3 network device) that your packet reaches on the way to the destination.

Mote: A Layer-3 device is a device that forwards packets based on IP addresses (Network Layer).

•If the output ends at hop **10**, that means traceroute reached the destination in **10 hops**.

•If the output ends at hop **14**, that means traceroute reached the destination in **14 hops**.

Note: Hop count is not a perfect "distance" measure, but it is a useful indicator of how many networks/devices your traffic crosses.

3) Why there are three times (three RTT values) per hop

Traceroute sends multiple probes (usually **3**) to each hop.
That is why you see three different time values such as:

8 ms 15 ms 11 ms

These values are **round-trip times (RTT)**:

•your probe goes to that router,

•the router sends a reply back,

•traceroute measures the total time.

The three values are often different because the Internet is dynamic:

•queueing delay changes,

•routers may prioritize traffic differently,

•congestion varies moment to moment.

4) What does "*" mean?

An asterisk * means that traceroute did not receive a reply for that probe within the timeout period.

This can happen for several reasons:

1.Packet loss (the probe or the reply was dropped somewhere).

2.Filtering / firewall rules (the router may block traceroute/ICMP replies).

3.Rate limiting (the router limits how frequently it replies to such probes).

4.Temporary congestion causing timeouts.

Key point:
If you see * but traceroute continues to later hops, it usually means "no reply from that router," not that the entire route is broken.

5) Understanding private vs public IP addresses in the output

You may see IP addresses from these private ranges:

•192.168.x.x

•10.x.x.x

•172.16.x.x to 172.31.x.x

Private IP addresses are not globally routable on the Internet.
When you see them in traceroute output, they typically indicate routers inside:

•your home network,

•your ISP's internal/private infrastructure,

•carrier-grade internal routing domains.

Examples:

•192.168.0.1 is usually your **home router / gateway**.

•10.x.x.x or 172.25.x.x often belongs to your **ISP's internal network**.

Public IP addresses (for example 94.54.156.1, 216.239.x.x, 104.20.x.x) are routable on the global Internet and often indicate:

•ISP edge routers,

•peering/transit points,

•destination network (or CDN) infrastructure.

6) Why different destinations have different hop counts

Two different websites can have very different routes because:

1.They are hosted in different networks (different ISPs/ASNs).

2.Your ISP has different peering and transit agreements for different destinations.

3.The destination may be behind a **CDN** (Content Delivery Network), meaning you are routed to a nearby edge server rather than the origin server.

4.Routing is based on **policy**, not only shortest path. Networks often choose routes based on cost, performance, and business agreements.

This is why one destination might be reached in 10 hops while another takes 14 hops.

7) Why the path can change even for the same destination

Even if you run traceroute to the same destination multiple times, the path might change slightly because of:

•**Load balancing** (traffic may be distributed across multiple equal-cost paths, called ECMP).

•**Dynamic routing updates** (routes change due to congestion, failures, or maintenance).

•**CDN decisions** (the DNS answer and edge selection may change depending on load and location).

•**Temporary network conditions**.

So, traceroute paths are often similar, but not guaranteed to be identical every time.

8) How to "read" the output like a network engineer

A practical interpretation approach:

1.Hop 1: usually your local gateway (home router).

2.Early hops (2–6): often your ISP access and internal backbone routers.

3.Middle hops: peering/transit networks or backbone segments.

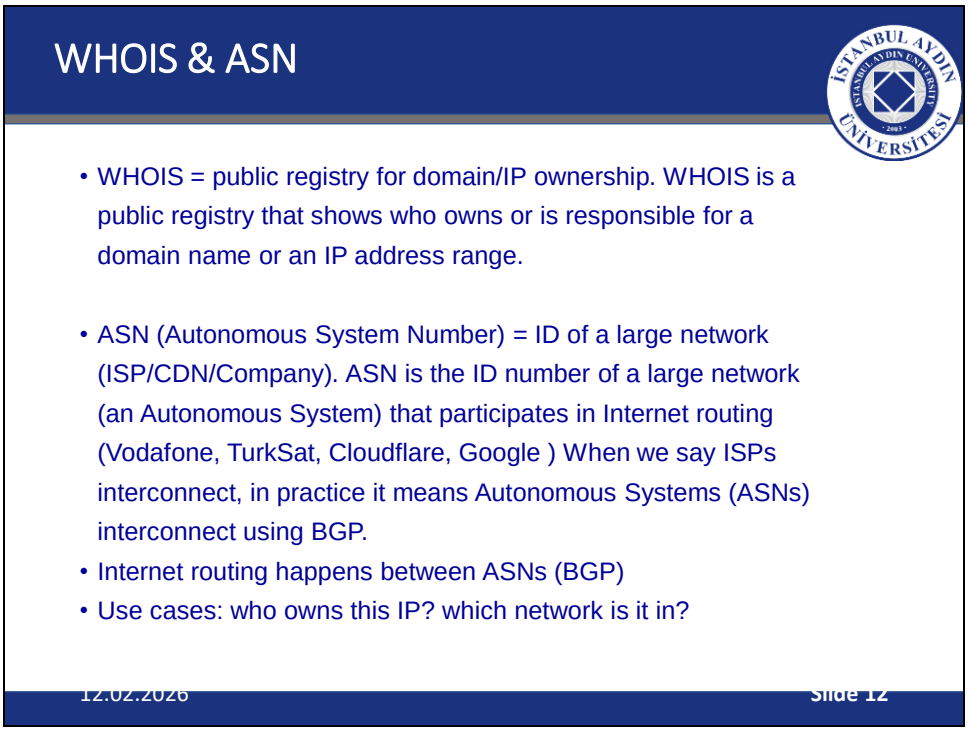
4.Last hops: destination network or CDN edge close to the destination server.

Look for:

•**Large jumps in RTT** (example: from ~20 ms to ~80 ms).
This may indicate:

- crossing a long geographic distance,
- entering a different major network,
- a congested link or router.

•**Repeated router names or IPs**: sometimes the same device appears again due to how probes are handled or due to load balancing.



WHOIS is a public registry that shows who owns or is responsible for a domain name or an IP address range.

ASN = Autonomous System Number

WHOIS is a public registry system that helps us find out **who is responsible for a domain name or an IP address range**. When you see a website or an IP address on the Internet, WHOIS is one of the first places to check **ownership and administrative responsibility**.

There are two common types of WHOIS-style lookups:

1.Domain WHOIS (for example, cloudflare.com)
This can show the domain registrar, registration dates, and administrative contacts (sometimes privacy-protected).

2.IP WHOIS / IP allocation records
This tells us which organization (ISP, company, institution) a block of IP addresses is assigned to.

In modern Internet routing, the key concept is the **Autonomous System (AS)**. An Autonomous System is a large network (for example, Vodafone, TurkSat, Google, Cloudflare) that manages its own routing policy. Each AS has a unique identifier called an **ASN (Autonomous System Number)**.

Why is ASN important?

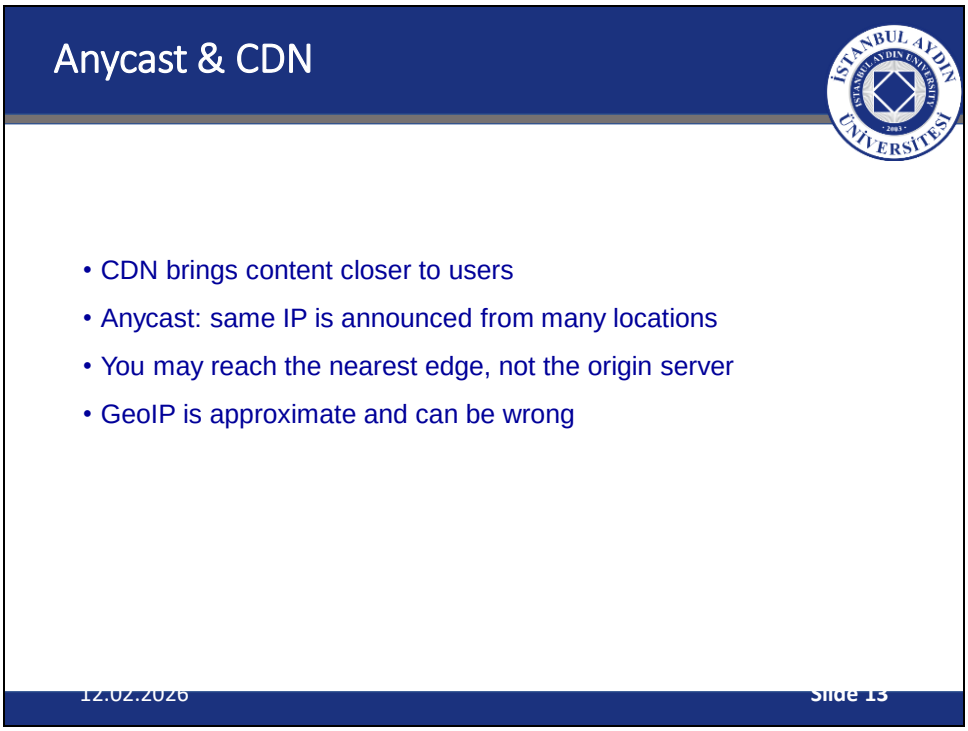
Because on the global Internet, routing decisions are primarily made **between ASNs** using a protocol called **BGP (Border Gateway Protocol)**.
So when we say “ISPs interconnect,” in practice we mean:
Autonomous Systems (ASNs) interconnect and exchange routes using BGP.

Practical interpretation:

•If an IP belongs to ASN 13335, it likely belongs to **Cloudflare**.

•If an IP belongs to an ISP ASN, it likely belongs to **Vodafone**, **TurkSat**, etc.
This helps us understand *which networks our traffic travels through*.

Ethical note (important):
This is **open-source information** used for understanding Internet structure, troubleshooting, and security awareness. It is not hacking or attacking.



Many modern websites do not serve content directly from a single "origin server." Instead, they use a **CDN (Content Delivery Network)**. A CDN places servers in many geographic locations so that users can receive content from a nearby server. This reduces delay and improves performance.

A key technology used by many CDNs is **Anycast addressing**.

Anycast means:
The same IP address can be advertised from multiple locations on the Internet. When you connect to that IP, Internet routing will typically deliver your traffic to the **nearest or best** CDN edge location according to routing policies.

This leads to an important result:

•The IP address you see in tracert might belong to a CDN (like Cloudflare).

•You are often reaching a **CDN edge server**, not the actual university/company origin server.

That is why IP "location" websites can be misleading.
They usually use **GeoIP databases**, which provide an estimate of location based on registration data, routing hints, or ISP information. But for Anycast IPs, the same IP can serve many cities/countries, so a GeoIP site may show a default or administrative location rather than the real physical edge you reached.

How to interpret results correctly:

•If the IP is labeled **CDN** and **Anycast**, treat location as “approximate / possibly misleading.”

•Focus on **who owns the IP (ASN/ISP/CDN)** and what role it plays (edge, transit, ISP core).

•In traceroute, a CDN IP near the end often indicates the destination is protected and accelerated by a CDN.

This is very useful in real life:

•troubleshooting performance

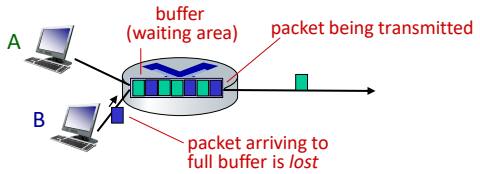
•understanding web infrastructure

•basic security awareness (CDN provides DDoS protection, WAF, etc.)

1.4.3 End-to-End Delay

Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



* Check out the Java applet for an interactive animation on queuing and loss

In theory, we sometimes assume an infinite buffer, but real routers have **finite** buffers.

If a packet arrives when the buffer is full, the router drops it—this is packet loss.

From an end-system viewpoint, a lost packet looks like it was sent into the network but never arrived.

As traffic intensity increases, the probability of packet loss increases.

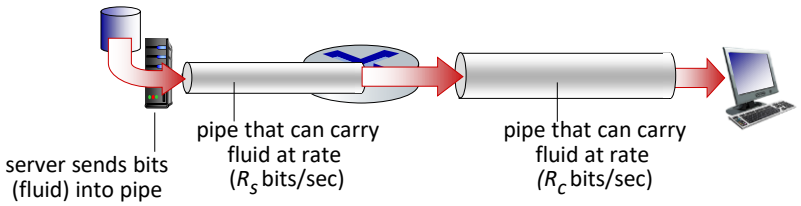
Whether a lost packet is retransmitted depends on the protocol.

For example, TCP may retransmit to ensure delivery; some real-time applications may choose not to retransmit.

1.4.4 Throughput in Computer Networks

Throughput

- *throughput:* rate (bits/time unit) at which bits are being sent from sender to receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time



Now we introduce another performance measure: **throughput**.

Throughput is the rate at which bits are delivered from sender to receiver.

We can talk about:

•**Instantaneous throughput** at a moment in time

•**Average throughput** over a longer period

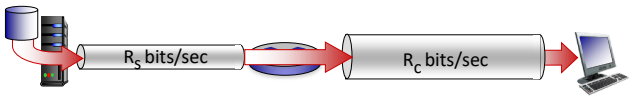
If a file has F bits and takes T seconds to download, the average throughput is **F/T bits per second**.

Different applications care differently: VoIP and real-time video need stable throughput above a threshold; file download mainly wants the highest possible average throughput.

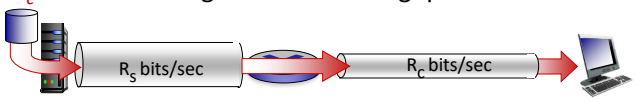
1.4.4 Throughput in Computer Networks

Throughput

$R_s < R_c$ What is average end-end throughput?



$R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

In an end-to-end path with multiple links, the end-to-end throughput is limited by the slowest link.

If the server can send at rate **Rs** and the client link is **Rc**, then end-to-end throughput is

approximately **min(Rs, Rc)**

If Rs is smaller, throughput is Rs.

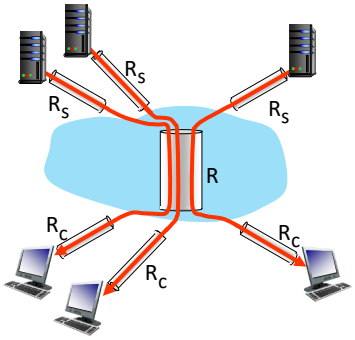
•If Rc is smaller, throughput is Rc.

The link that limits throughput is called the **bottleneck link**.

Think of bits as fluid and links as pipes: the narrowest pipe limits the flow.

1.4.4 Throughput in Computer Networks

Throughput: network scenario



10 connections (fairly) share backbone bottleneck link *R* bits/sec

- per-connection end-end throughput: *min(R_c,R_s,R/10)*
- in practice: R_c or R_s is often bottleneck

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/

In real networks, many flows share core links.

Here, ten connections share a common backbone link with capacity **R**.

If the link is shared fairly, each connection gets about **R/10**.

So the approximate per-connection throughput becomes:
min(Rc, Rs, R/10).

In practice, the access links Rs or Rc often become the bottleneck, but sometimes the shared core link becomes the bottleneck.

1.4.4 summarize

- To summarize, packet-switched networks introduce three key performance issues:
  delay, loss, and throughput.
- Delay comes from processing, queueing, transmission, and propagation.
  Loss happens when buffers overflow.
  Throughput depends on the bottleneck link and competing traffic.
- Understanding these concepts helps us explain why real networks sometimes feel slow, unstable, or inconsistent.

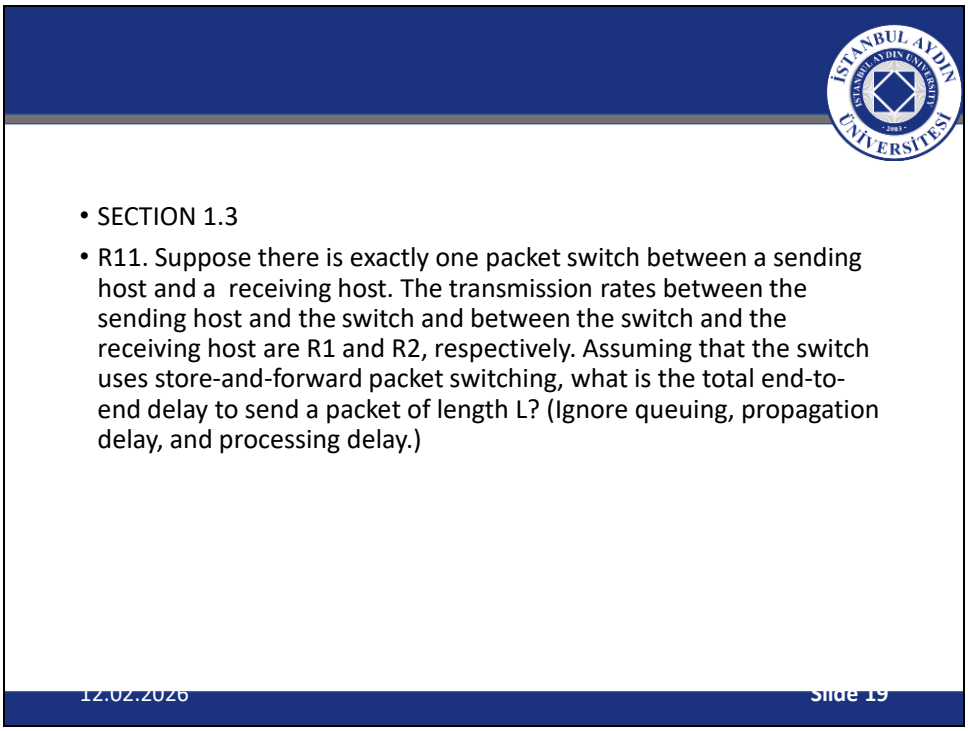
To summarize, packet-switched networks introduce three key performance issues: **delay, loss, and throughput**.

•Delay comes from processing, queueing, transmission, and propagation.

•Loss happens when buffers overflow.

•Throughput depends on the bottleneck link and competing traffic.

Understanding these ideas explains why real networks can feel slow, unstable, or inconsistent.



11. At time t0 the sending host begins to transmit. At time t1 = L/R1, the sending host

completes transmission and the entire packet is received at the router (no propagation

delay). Because the router has the entire packet at time t1, it can begin to transmit the

packet to the receiving host at time t1. At time t2 = t1 + L/R2, the router completes

transmission and the entire packet is received at the receiving host (again, no

propagation delay). Thus, the end-to-end delay is L/R1 + L/R2.

• SECTION 1.3

R13. Suppose users share a 2 Mbps link. Also suppose each user transmits continuously at 1 Mbps when transmitting, but each user transmits only 20 percent of the time.

a) When circuit switching is used, how many users can be supported?

b) For the remainder of this problem, suppose packet switching is used. Why will there be essentially no queuing delay before the link if two or fewer users transmit at the same time? Why will there be a queuing delay if three users transmit at the same time?

c) Find the probability that a given user is transmitting.

d) Suppose now there are three users. Find the probability that at any given time, all three users are transmitting simultaneously. Find the fraction of time during which the queue grows.

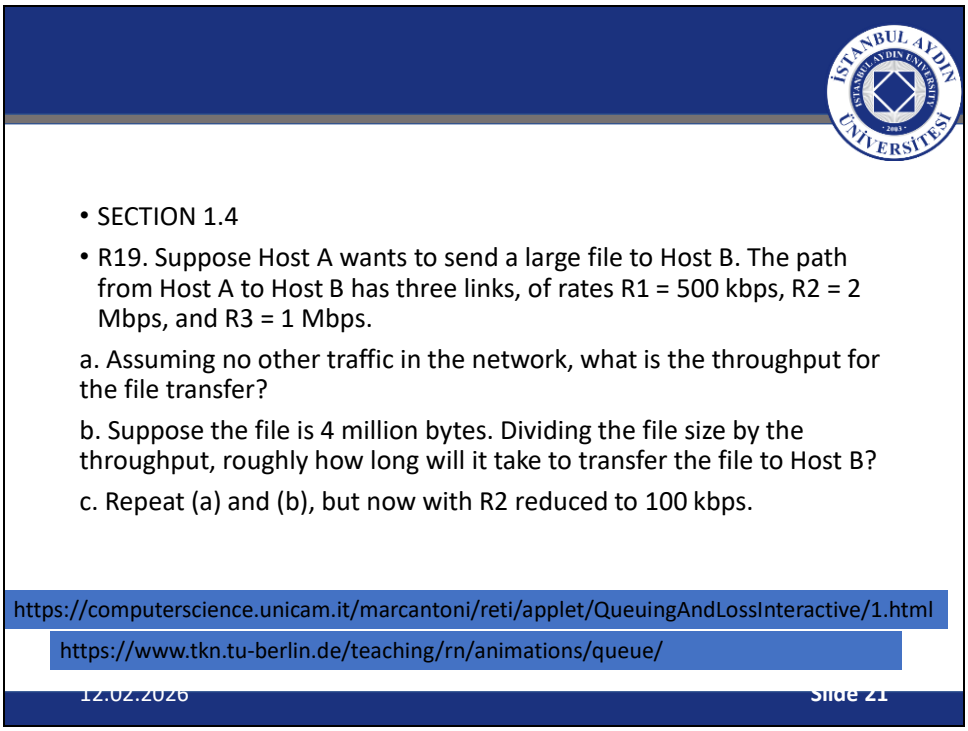
13. a) 2 users can be supported because each user requires half of the link bandwidth.

b) Since each user requires 1Mbps when transmitting, if two or fewer users transmit simultaneously, a maximum of 2Mbps will be required. Since the available bandwidth of the shared link is 2Mbps, there will be no queuing delay before the link. Whereas, if three users transmit simultaneously, the bandwidth required will be 3Mbps which is more than the available bandwidth of the shared link. In this case, there will be queuing delay before the link.

c) Probability that a given user is transmitting = 0.2

d) Probability that all three users are transmitting simultaneously = $((3,3)\ p^3(1-P)^{3-3} = (0.2)^3 = 0.008$.

Since the queue grows when all the users are transmitting, the fraction of time during which the queue grows (which is equal to the probability that all three users are transmitting simultaneously) is 0.008.



Important Idea: Bottleneck Link

When a file travels through multiple links, the overall throughput is limited by the **slowest link** on the path. This slowest link is called the **bottleneck link**.

So, in general:

Throughput=min(R1,R2,R3)

This means we compare the link speeds and choose the smallest one.

(a) What is the throughput for the file transfer?

The three link rates are:

•R1=500 kbps

•R2=2 Mbps = 2000 kbps

•R3=1 Mbps = 1000 kbps

Now compare them:

min(500, 2000, 1000)=500 kbps

Answer for (a):

Throughput=500 kbps

Explanation:

Even though the second and third links are faster, the first link can only send

data at **500 kbps**. Therefore, the entire file transfer cannot go faster than **500 kbps**.

(b) If the file is 4 million bytes, how long will it take to transfer?

The file size is given in **bytes**, but throughput is given in **bits per second**, so first we must convert bytes to bits.

Step 1: Convert file size to bits

4,000,000 bytes×8=32,000,000 bits So the file size is:

32,000,000 bits

Step 2: Convert throughput to bits per second

500 kbps=500,000 bits/second

Step 3: Use the formula

Time=File size/ Throughput

Time=32,000,000/500,000=64 seconds

Answer for (b):

64 seconds\boxed{64\ \text{seconds}}64 seconds

Explanation:

The network can deliver **500,000 bits every second**. Since the file contains **32,000,000 bits**, it takes:

32,000,000÷500,000=64

So the transfer time is **64 seconds**.

(c) Repeat (a) and (b), but now with R2=100R_2 = 100R2=100 kbps

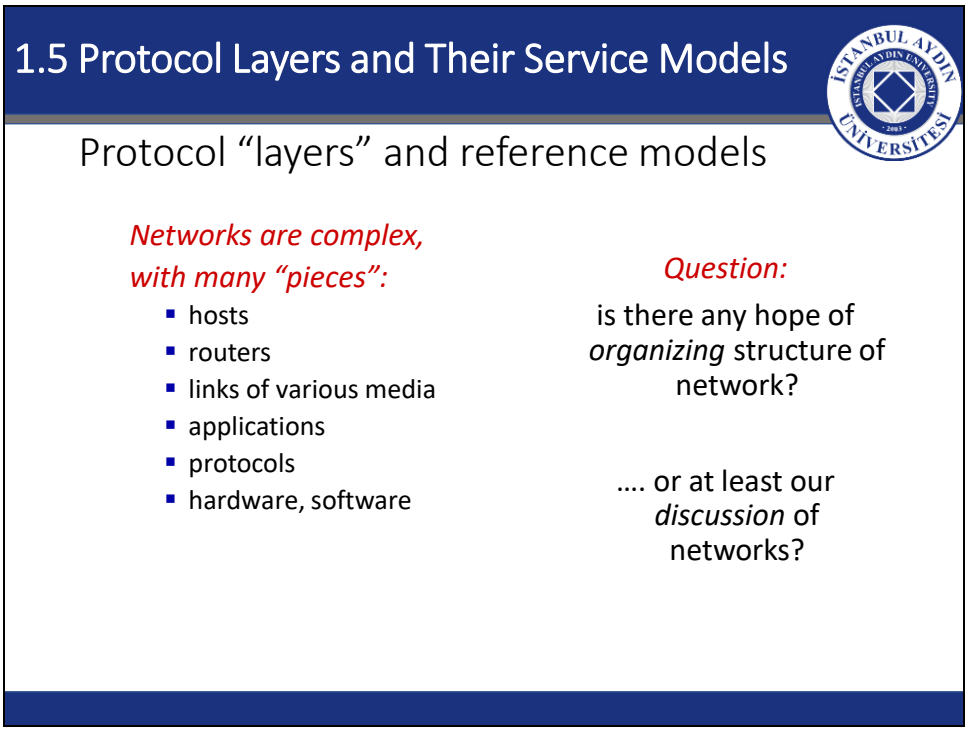
min(500, 100, 1000)=100 kbps

Throughput=100 kbps

320÷60≈5.33 minutes

Mini Quiz

- “Why would two ISPs choose settlement-free peering instead of paying a provider?”
- “Why do content providers build private networks instead of using only ISPs?”
- “What happens if an ISP loses one upstream provider? (multi-homing idea)”



At this point, it is clear that the Internet is a very complex system.

We have many “pieces”: hosts, routers, links with different media, applications, protocols, and both hardware and software.

So the key question is: **Is there any hope of organizing this complexity?**

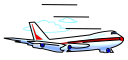
Fortunately, yes. We use **layers** and **reference models** to organize our understanding and design.

A layered approach gives us a structured way to talk about networks.

Instead of trying to understand everything at once, we study one layer at a time and focus on what service each layer provides.

1.5.1 Layered Architecture

Example: organization of air travel



ticket (purchase)
baggage (check)
gates (load)
runway takeoff
airplane routing

ticket (complain)
baggage (claim)
gates (unload)
runway landing
airplane routing

airplane routing

airline travel: a series of steps, involving many services

To understand layering, we can use an analogy: **airline travel**.

Think about what happens when you fly:

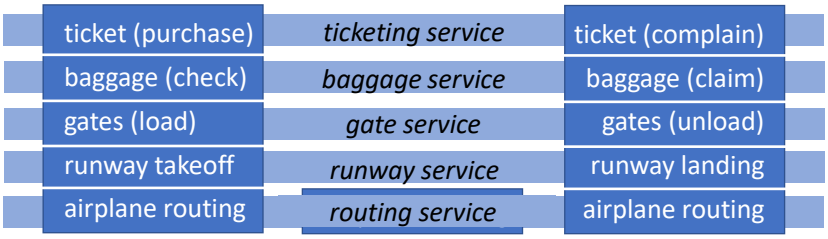
you buy a ticket, check baggage, go to the gate, take off, get routed in the air, land, unload, claim baggage, and possibly complain.

This system is complex, but we can understand it because it is organized into clear steps and roles.

Networks are similar: they are complex, but we can organize them with a layered structure.

1.5.1 Layered Architecture

Example: organization of air travel



layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

Q: describe in words the service provided in each layer above

Now we add the key idea: each layer provides a **service**.

In the airline example, we can group actions into layers like ticketing service, baggage service, gate service, runway service, and routing service.

The key idea is: **each layer provides a service**.

1. performing actions inside the layer, and
2. using the service of the layer below.

For example, the gate service depends on the runway service.

Similarly, in networking, one layer depends on the layer below it.

Key takeaway: Layers hide internal complexity and provide a clean service interface to the layer above.

1.5.1 Layered Architecture

Why layering?

dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change in layer's service *implementation*: transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- layering considered harmful?
- layering in other complex systems?

Why do we use layering?

Layering has two major benefits:

1.Explicit structure: It helps us identify the parts of a complex system and describe how they relate.

it gives us an explicit structure to discuss complex systems.

1.Modularity: It becomes easier to maintain and update the system.

2.If we change how a layer is implemented, the rest of the system does not need to change—as long as the service stays the same

it provides modularity.

If we change how one layer is implemented, the rest of the system does not need to change—as long as the service stays the same.

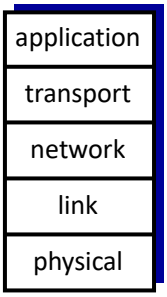
For example, if an airport changes the gate procedure, the whole airline system still works.

The book also mentions that some engineers criticize layering, because sometimes layers can duplicate functions or need information from other layers.
But overall, layering is extremely useful.

1.5.1 Layered Architecture

Internet protocol stack

- *application:* supporting network applications
  - IMAP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi), PPP
- *physical:* bits "on the wire"



The Internet uses a five-layer protocol stack:
Application, Transport, Network, Link, and Physical.

Each layer has a specific role:

•**Application layer:** where network applications and application protocols live (HTTP, SMTP, DNS).

•**Transport layer:** moves application messages between processes (TCP/UDP). TCP can provide reliable delivery and flow control; UDP is a simpler, connectionless service.

•**Network layer:** moves datagrams host-to-host (IP) and uses routing protocols to choose paths. IP is often called the "glue" that binds the Internet together.

•**Link layer:** moves frames between neighboring nodes on a single link (Ethernet, WiFi, PPP).

•**Physical layer:** moves individual bits over the medium (copper, fiber, radio).

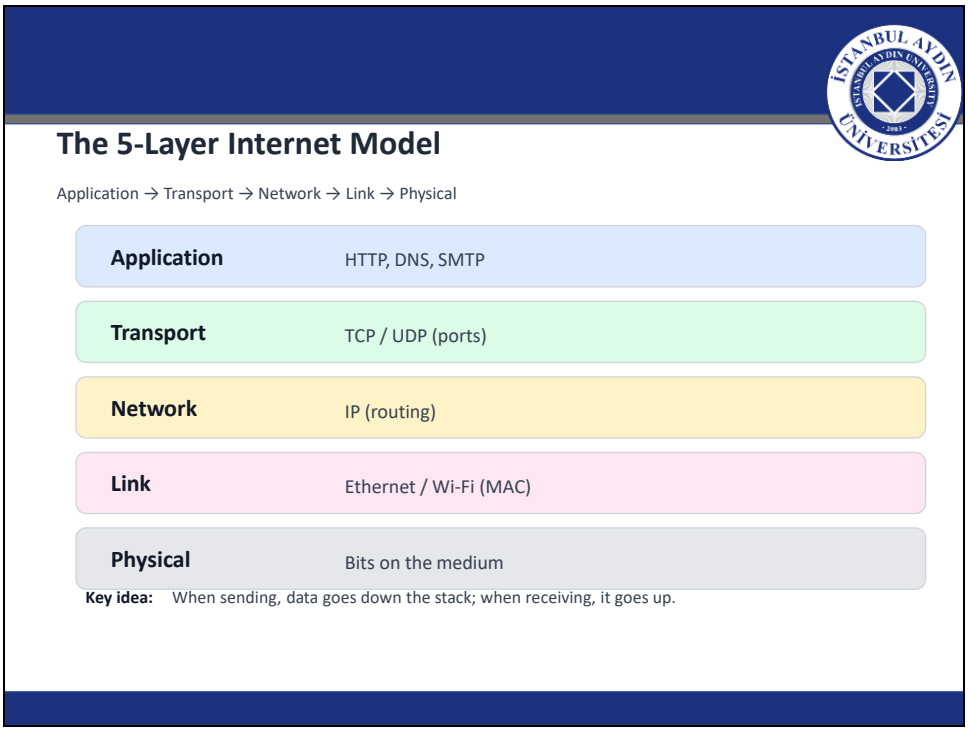
Key takeaway: Each layer has a clear job, and together they form the protocol stack.

Encapsulation in Networking

How data is wrapped as it moves down the protocol stack (sender) and unwrapped at the receiver.

- Each layer adds its own header (control information).
- The payload of a layer is usually the packet from the layer above.
- This “wrapping” is called encapsulation.
- At the receiver, the process is reversed (decapsulation).

Course note (Computer Networks)



•Each layer adds its own header (control information).

•The payload of a layer is usually the packet from the layer above.

•This “wrapping” is called encapsulation.

•At the receiver, the process is reversed (decapsulation).

•Application — HTTP, DNS, SMTP

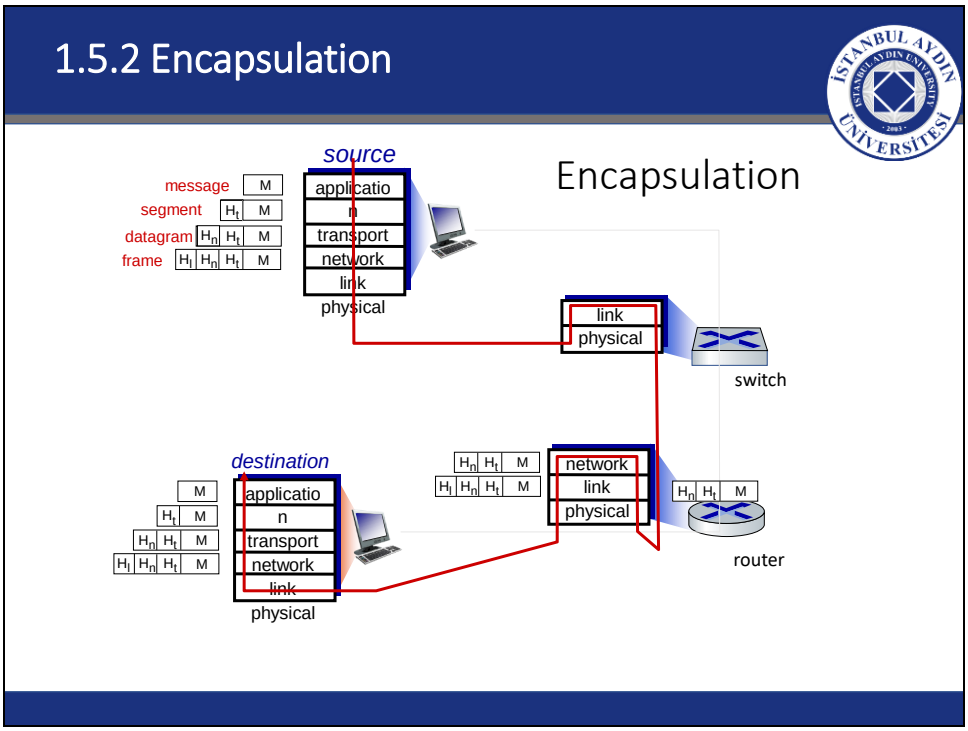
•Transport — TCP / UDP (ports)

•Network — IP (routing)

•Link — Ethernet / Wi-Fi (MAC)

•Physical — Bits on the medium

Key idea:
When sending, data goes down the stack; when receiving, it goes up.



Encapsulation is one of the most important concepts in networking.

Data moves down the stack at the sender and up the stack at the receiver.

Start with an application-layer **message (M)**.

The transport layer adds a transport header **Ht**, creating a **segment**.
Then the network layer adds a network header **Hn**, creating a **datagram**.
Then the link layer adds a link header **Hl**, creating a **frame**. So each layer "wraps" the data with its own header.
That is why we call it encapsulation.

So at each layer, a packet has:

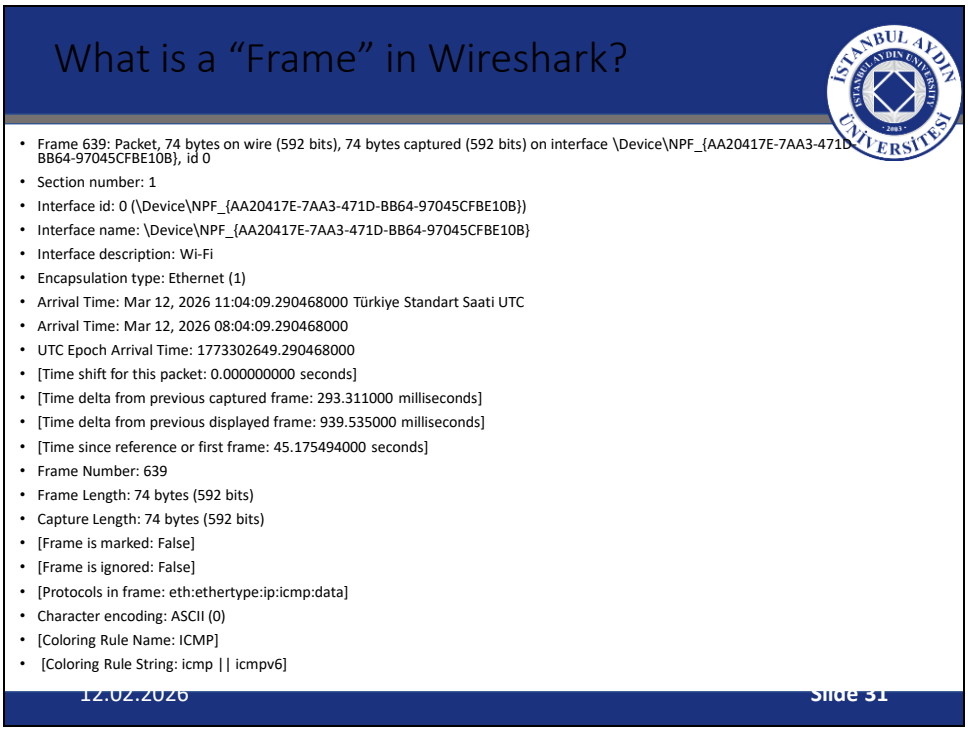
•**Header fields** (control information), and

•a **payload** (usually the packet from the layer above).

The added header information can include addresses, information needed for delivery to the correct application, and error-detection bits.

Another key point from this figure:
hosts implement all five layers, but routers typically implement only the lower layers, and link-layer switches implement even fewer layers.

This supports an important design principle:
much of the Internet's complexity is placed at the edges.



What You Are Looking At (Frame 639)
Wireshark captured **one frame** on the **Wi-Fi** interface.

•A **frame** is the **Layer-2 container** that carries Layer-3 and above.

•Protocol stack in this frame: **eth → ip → icmp → data**

•Frame size: **74 bytes (592 bits)** “on the wire”

•“Frame is not the same as IP packet. A frame is what is actually sent over the local link. Inside the frame we find an IP packet, and inside IP we find ICMP.”

Layer 2: Ethernet II (Link Layer)

- Ethernet II, Src: Intel_92:b7:b5 (d4:25:8b:92:b7:b5), Dst: 0a:7e:32:5b:32:60 (0a:7e:32:5b:32:60)
- Destination: 0a:7e:32:5b:32:60 (0a:7e:32:5b:32:60)
- .... ..1. .... = LG bit: Locally administered address (this is NOT the factory default)
- ....0 .... = IG bit: Individual address (unicast)
- Source: Intel_92:b7:b5 (d4:25:8b:92:b7:b5)
- .... ..0. .... = LG bit: Globally unique address (factory default)
- ....0 .... = IG bit: Individual address (unicast)
- Type: IPv4 (0x0800)
- [Stream index: 0]

12.02.2026 Slide 32

•Purpose: deliver the frame to the **next hop** on the local network.

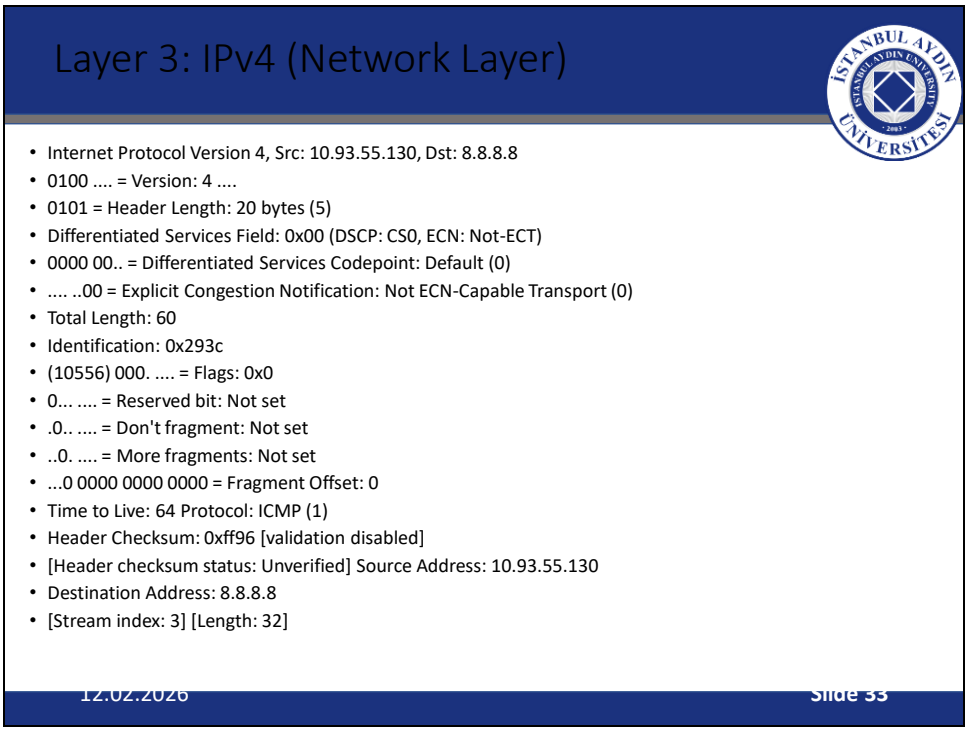
•Key fields (from the capture):

- **Source MAC** = your NIC (wireless card)
- **Destination MAC** = typically the **gateway/router** (next hop)
- **EtherType 0x0800** = payload is **IPv4**

•Important rule:

- **MAC addresses are local (hop-by-hop)**
- **IP addresses are end-to-end**

"Students often think the destination MAC should be the final destination. In most cases it's the gateway's MAC, because MAC is only for the local link. The IP destination is the real end-to-end target."



•Purpose: route the packet to the **final destination across networks**

•Key fields (from the capture):

- **Src IP:** 10.93.55.130 (your host)
- **Dst IP:** 8.8.8.8 (Google DNS)
- **TTL:** 64 (decreases by 1 at every router → prevents loops)
- **Protocol:** ICMP (1) → next header is ICMP (not TCP/UDP)

•**Total Length:** 60 bytes (IP header + ICMP + data)

"I point to TTL and explain hop count. Then I point to the Protocol field: it tells what comes after IP. Here it is ICMP, so there will be no TCP or UDP header."

ICMP: Ping (Echo Request

- Internet Control Message Protocol
- Type: Echo (ping) request (8)
- Code: 0
- Checksum: 0x0118 [correct]
- [Checksum Status: Good]
- Identifier (BE): 1 (0x0001)
- Identifier (LE): 256 (0x0100)
- Sequence Number (BE): 19523 (0x4c43)
- Sequence Number (LE): 17228 (0x434c) [Response frame: 640] Data (32 bytes)
- Data: 61626364656667686 96a6b6c6d6e6f707172737475767761626364656667 6869
- [Length: 32]

12.02.2026 Slide 34

•ICMP is used for **network control and diagnostics** (e.g., ping).

•In this packet:

- **Type 8** = Echo Request
- **Code 0** = standard request
- **Identifier + Sequence Number** = match requests with replies and track order
- **Checksum** = error detection for ICMP message

•Wireshark indicates the matching **Echo Reply** frame (response)

"I explain: ping sends a request and expects a reply. Wireshark even points to the response frame number. That's how we measure reachability and RTT."

Why There Is NO Transport Header Here

- Many packets: **IP → TCP/UDP → Application**
- But this packet: **IP → ICMP**
- Because IPv4 **Protocol = ICMP (1)**
- So **no TCP header, no UDP header, no ports** in this packet
- “This is a great moment to correct a common misconception: ‘All Internet traffic uses TCP/UDP.’ Not true. ICMP is directly on top of IP.”

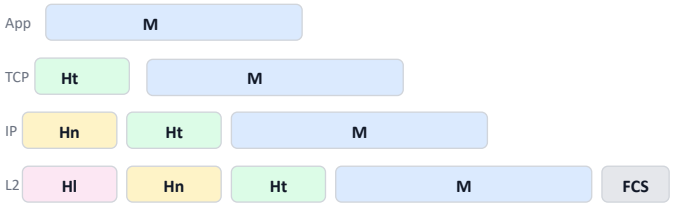
Data (Payload) and Sizes

- ICMP includes **32 bytes of data** (test payload).
- Size relationship (from your capture):
  - **Ethernet frame length:** 74 bytes
  - **IP total length:** 60 bytes
  - Explanation: 14 bytes Ethernet header + 60 bytes IP packet = 74 bytes
- The Physical layer transmits all of this as **bits** on the medium.
- "I show the math quickly. This connects encapsulation to real byte counts: the outer header adds overhead."

Encapsulation: What are M, Ht, Hn, Hl?

They are labels used in diagrams (not commands). Each header is added by a different layer.

- M = Application-layer Message (the actual content: HTTP request, email text, etc.)
- Ht = Transport header (TCP/UDP: source/destination ports, reliability fields)
- Hn = Network header (IP: source/destination IP, TTL, protocol)
- Hl = Link header (Ethernet/Wi-Fi: source/destination MAC, frame type)



At the receiver, headers are removed in the reverse order (decapsulation).

Concrete Example: HTTP over TCP/IP over Ethernet

This is what the payload “looks like” inside each layer.

Application (M): HTTP message

GET /index.html HTTP/1.1
Host: example.com
User-Agent: MyBrowser/1.0
Accept: text/html

Transport (Ht): TCP header (selected fields)

Src Port: 51000
Dst Port: 80
Seq: 1001
ACK: 5001
Flags: PSH, ACK

Network (Hn): IP header (selected fields)

Src IP: 192.168.1.10
Dst IP: 93.184.216.34
TTL: 64
Protocol: TCP (6)

Link (Hl): Ethernet header + trailer

Dst MAC: 00:1A:2B:3C:4D:5E (gateway)
Src MAC: 70:88:6B:AA:BB:CC
EtherType: 0x0800 (IPv4)
FCS/CRC: error-detection bits

Physical layer: the frame is transmitted as bits (0s and 1s) on the medium.

Who Implements Which Layers?

End hosts implement all layers; network devices often implement fewer layers.

- Hosts (PCs/phones/servers): Application + Transport + Network + Link + Physical
- Routers: typically Network + Link + Physical (forward packets based on IP)
- Link-layer switches: typically Link + Physical (forward frames based on MAC)
- Design principle: keep the core simple; put more intelligence at the edges.

Quick Check (Self-Study)

Answer these to confirm you understand encapsulation.

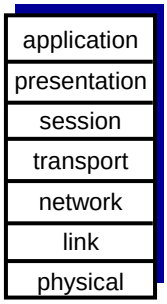
- What is the payload at the IP layer?
- Which header contains port numbers: Ht, Hn, or Hl?
- Which header contains IP addresses?
- Why does a switch not need to read the TCP header to forward a frame?
- If the destination MAC changes at each hop, which header changes: Hl, Hn, or Ht?

1.5.2 Encapsulation

ISO/OSI reference model

Two layers not found in Internet protocol stack!

- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - needed?



The seven layer OSI/ISO reference model

There is also a seven-layer reference model called OSI.

OSI is a **seven-layer** reference model. It includes two layers that are not explicitly shown in the Internet's five-layer stack:

•**Presentation layer:** helps applications interpret data (encryption, compression, data formats).

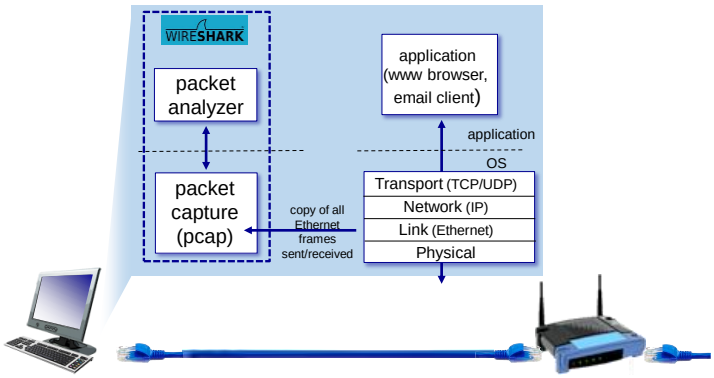
•**Session layer:** manages synchronization, checkpointing, and recovery of data exchange.

In the Internet architecture, these functions often appear inside applications when needed.

OSI is mainly a teaching/reference model, but it helps us organize ideas and compare architectures.

1.5.2 Encapsulation

Wireshark



Finally, Wireshark helps us *see* these layers in real traffic.

Wireshark allows us to **see the layers in real traffic**. It captures packets and shows the headers at different layers, such as:

•Ethernet at the link layer

•IP at the network layer

•TCP/UDP at the transport layer

This is extremely useful because it connects theory with reality. When you see a packet in Wireshark, you can literally identify encapsulation and the protocol stack.

1.6 Networks Under Attack

Network Security: "What can go wrong?"

- field of network security:
  - how bad guys can attack computer networks
  - how we can defend networks against attacks
  - how to design architectures that are immune to attacks
- Internet not originally designed with (much) security in mind
  - *original vision:* "a group of mutually trusting users attached to a transparent network" ☺
  - Internet protocol designers playing "catch-up"
  - security considerations in all layers!

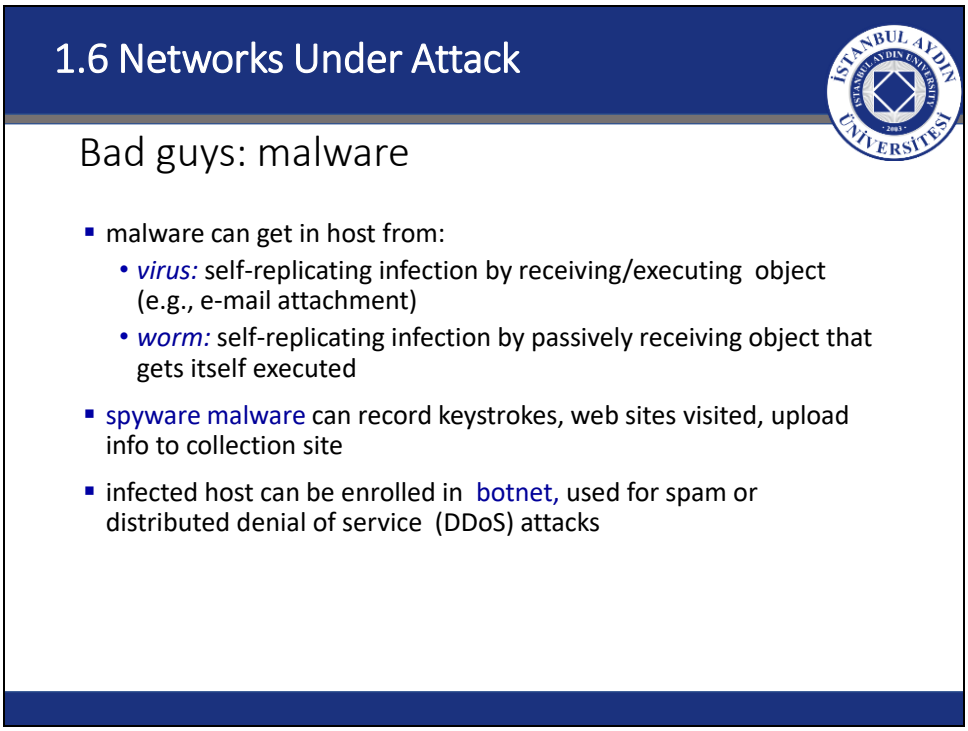
The Internet has become mission critical for companies, universities, governments, and individuals.

At the same time, there is a "dark side": attackers try to damage systems, steal private information, and disrupt services we depend on.

Network security is the field that studies two questions:

1.How attackers can exploit computer networks, and

2.How we can defend networks — or design architectures that are resistant to attacks.



One major threat is **malware** — malicious software that can enter and infect a host via the Internet

Once malware infects a device, it can do many harmful actions, such as:

•deleting files,

•installing spyware that collects sensitive information (passwords, keystrokes), and

•sending this information back to attackers.

A compromised host can also become part of a **botnet** — a large network of infected devices controlled by an attacker.

Botnets are often used to send spam or launch distributed denial-of-service attacks.

Key takeaway: Malware turns a normal computer into a tool that attackers can control.

1.6 Networks Under Attack

Self-replicating malware (spreads exponentially)

- malware can get in host from:
  - *virus:* self-replicating infection by receiving/executing object (e.g., e-mail attachment)
  - *worm:* self-replicating infection by passively receiving object that gets itself executed
- spyware malware can record keystrokes, web sites visited, upload info to collection site
- infected host can be enrolled in botnet, used for spam or distributed denial of service (DDoS) attacks

Text to place under the slide:

Much of today's malware is self-replicating.

After infecting one host, it tries to infect other hosts across the Internet.

Those newly infected hosts then infect even more hosts, creating very fast, exponential spread.

This is why outbreaks can become global in a short time if users and systems are not protected.

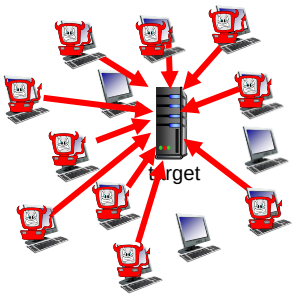
Practical message: Updates, antivirus, and cautious behavior are not "optional"; they are part of basic digital safety.

1.6 Networks Under Attack

Bad guys: denial of service

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts

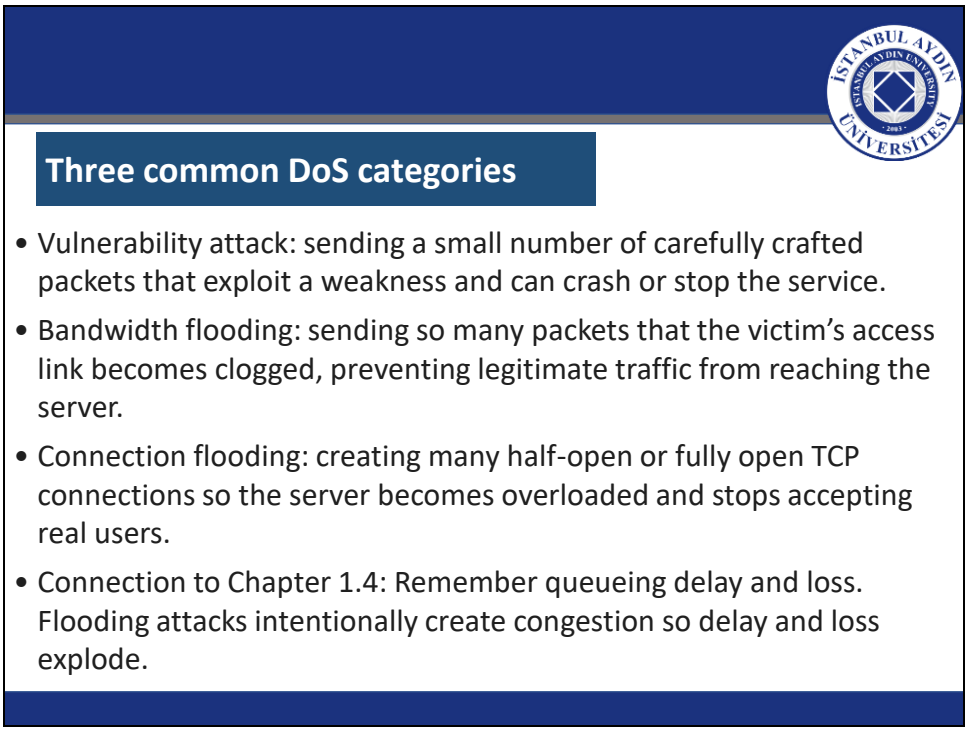


Another broad class of threats is **denial-of-service (DoS)** attacks.

The goal of a DoS attack is to make a network, server, or service unusable for legitimate users.

DoS attacks commonly target web servers, email servers, DNS servers, and even entire institutional networks.

The key idea is simple:
Even if a system works perfectly under normal traffic, it may fail if attackers can force it to handle too much traffic or too many connections.



Three common DoS categories (vulnerability, bandwidth flooding, connection flooding)

Most DoS attacks fall into three categories:

Vulnerability attack: sending a small number of carefully crafted packets that exploit a weakness and can crash or stop the service.

Bandwidth flooding: sending so many packets that the victim's access link becomes clogged, preventing legitimate traffic from reaching the server.

Connection flooding: creating many half-open or fully open TCP connections so the server becomes overloaded and stops accepting real users.

Connection to Chapter 1.4: Remember queueing delay and loss. Flooding attacks intentionally create congestion so delay and loss explode.

DoS vs DDoS (and Botnets)

- DoS: one attacker tries to exhaust server or network resources
- DDoS: many compromised devices attack the same target together
- Botnet: infected hosts controlled by an attacker (used for DDoS / spam)
- Why it matters: harder to block, traffic comes from many sources

• DoS (Denial of Service) aims to make a service unavailable by overwhelming a resource: CPU, memory, bandwidth, or connection slots.

• DDoS is the same idea but launched from many machines at once, often using a botnet.

• Connect to Chapter 1.4: congestion creates queueing delay and packet loss—DDoS attacks intentionally create congestion.

Quick question (30 seconds): Why is DDoS harder to stop than a single-source DoS?

DDoS and botnets

- If a single attacker cannot generate enough traffic, attackers can launch a distributed DoS (DDoS) attack.
- In DDoS, the attacker controls many infected devices and orders them to send traffic toward a single target.
- Because traffic comes from many sources, DDoS is harder to detect and block than an attack from one host.
- Key takeaway: DDoS is powerful because it uses the “scale of the Internet” against the victim.

If a single attacker cannot generate enough traffic, attackers can launch a distributed DoS (DDoS) attack. In DDoS, the attacker controls many infected devices and orders them to send traffic toward a single target.

Because the traffic comes from many sources, DDoS is harder to detect and block than an attack from one host. Botnets with thousands of compromised hosts are commonly used for DDoS.

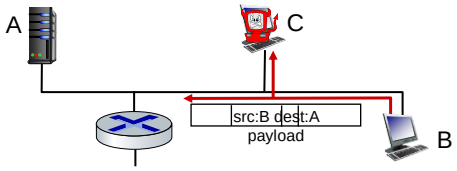
Key takeaway: DDoS is powerful because it uses the “scale of the Internet” against the victim.

1.6 Networks Under Attack

Bad guys: packet interception

packet "sniffing":

- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



Wireshark software used for our end-of-chapter labs is a (free) packet-sniffer

Many users access the Internet through wireless networks (WiFi or cellular).

This creates a vulnerability: a passive receiver nearby can capture copies of packets transmitted over the air.

A device or program that records packets is called a **packet sniffer**.

Sniffing can also happen in wired environments, especially in broadcast-style networks, or if an attacker gains access to a router or an access link.

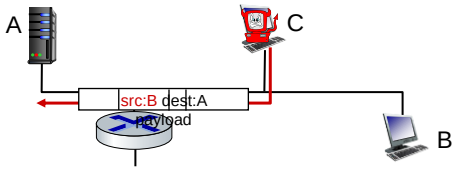
A key danger is that packets may contain sensitive information: passwords, private messages, or confidential data.

Because sniffers are passive, they are difficult to detect. One of the best defenses is **cryptography**, which we study more deeply in advanced security topics.

1.6 Networks Under Attack

Bad guys: fake identity

IP spoofing: send packet with false source address



… lots more on security (throughout, Chapter 8)

Attackers can also **masquerade** as someone you trust.

It is possible to create and send packets with a fake source address and arbitrary contents.

The Internet will still forward those packets to the destination.

Creating packets with a false source address is called **IP spoofing**.

This is one way an attacker can pretend to be another user or another machine.

To defend against masquerading, networks and applications need **end-point authentication** — mechanisms to verify that a message really comes from who it claims to come from.

Key takeaway: "Declared identity" is not the same as "verified identity."

Why the Internet Is Insecure (Big Picture)

- A very important historical point is that the Internet was originally designed for a world of "mutually trusting users."
- Security was not the central goal from the beginning.
- Many design choices reflect that trust model: users can send packets to other users, and identity is often taken "at face value."
- But today, Internet users are not mutually trusting—so security must be treated as a core requirement.

A very important historical point is that the Internet was originally designed for a world of "mutually trusting users."

Security was not the central goal from the beginning.

Many design choices reflect that trust model: by default, users can send packets to other users, and identity is often taken "at face value" rather than authenticated by default.

But today, Internet users are not mutually trusting. We often communicate with strangers, through third parties, and sometimes over untrusted networks. That is why modern networking must treat security as a core requirement.

Wrap-up: what we should remember

- Malware can infect hosts and create botnets.
- DoS/DDoS can disrupt services by overwhelming resources.
- Sniffing can capture sensitive information, especially over wireless.
- Spoofing can fake identity, so authentication matters.
- Main message: The Internet works amazingly well, but it also creates risks.

In this short security overview, we learned what can go wrong:

Malware can infect hosts and create botnets.

DoS/DDoS can disrupt services by overwhelming resources.

Sniffing can capture sensitive information, especially over wireless.

Spoofing can fake identity, so authentication matters.

Main message: The Internet works amazingly well, but it also creates risks. Understanding these risks helps us become better designers and users of networked systems.