Computer Networks

Ankit Singla

Some slides adapted from Brighten Godfrey, Jennifer Rexford, Scott Shenker, Laurent Vanbever
Computer Networks

Part 1: Overview

#1 What is a network made of?

#2 How is it shared?

#3 How does communication happen?

#4 How do we characterize it?
The Internet should allow

processes on different hosts
to exchange data

everything else is just commentary…
How do you exchange data in a network as complex as this?
To exchange data, Alice and Bob use a set of network protocols
A protocol is like a conversational convention: who should talk next and how they should respond.

Bob

Alice

hello

hello

give me http://ndal.ethz.ch/

here it is
Sometimes implementations are not compliant…

Alice

Bob

hello

give me http://...
give me http://...
give me http://...
give me http://...
give me http://...
give me http://...
give me http://...
Each protocol is governed by a specific interface

```plaintext
while (...) {
  message = ...;
  send(message, ...);
}
```

```plaintext
while (...) {
  message = receive(...);
}
```

Alice

WoW server

WoW client

Bob

Application Programming Interface
In practice, there exist a lot of network protocols. How does the Internet organize this?
How standards proliferate:
(See: A/C chargers, character encodings, instant messaging, etc.)

**Situation:**
There are 14 competing standards.

**14?! Ridiculous!**
We need to develop one universal standard that covers everyone’s use cases.

**Yeah!**

**Soon:**

**Situation:**
There are 15 competing standards.

https://xkcd.com/927/
Modularity is a key component of any good system.

Problem
- Can’t build large systems out of spaghetti code.
- Hard (if not, impossible) to understand, debug, update.
- Need to bound the scope of changes.
- Evolve the system without rewriting it from scratch.

Solution
- Modularity is how we do it.
- …and understand the system at a higher-level.
Modularity, based on abstraction, is the way things get done.

— Barbara Liskov, MIT

Photo: Donna Coveney
To provide structure to the design of network protocols, network designers organize protocols in layers which the network “stack” implements.
Internet communication can be decomposed in **5 independent layers** (or 7 layers for the OSI model)

- **L5**: Application
- **L4**: Transport
- **L3**: Network
- **L2**: Link
- **L1**: Physical
Each layer provides a service to the layer above

<table>
<thead>
<tr>
<th>layer</th>
<th>service provided</th>
</tr>
</thead>
<tbody>
<tr>
<td>L5</td>
<td>Application: network access</td>
</tr>
<tr>
<td>L4</td>
<td>Transport: end-to-end delivery (reliable or not)</td>
</tr>
<tr>
<td>L3</td>
<td>Network: global best-effort delivery</td>
</tr>
<tr>
<td>L2</td>
<td>Link: local best-effort delivery</td>
</tr>
<tr>
<td>L1</td>
<td>Physical: physical transfer of bits</td>
</tr>
</tbody>
</table>
What if no reliable transport is provided?

Every application that needs reliability has to engineer it from scratch

• Programmer burden
• Much higher likelihood of bugs
• Wasteful effort
Each layer provides a service to the layer above by using the services of the layer directly below it.

- **Applications**
  - built on...

- **Reliable (or unreliable) transport**
  - built on...

- **Best-effort global packet delivery**
  - built on...

- **Best-effort local packet delivery**
  - built on...

- **Physical transfer of bits**
Each layer has a unit of **data**

<table>
<thead>
<tr>
<th>layer</th>
<th>role</th>
</tr>
</thead>
<tbody>
<tr>
<td>L5 Application</td>
<td>exchanges <em>messages</em> between processes</td>
</tr>
<tr>
<td>L4 Transport</td>
<td>transports <em>segments</em> between end-systems</td>
</tr>
<tr>
<td>L3 Network</td>
<td>moves <em>packets</em> around the network</td>
</tr>
<tr>
<td>L2 Link</td>
<td>moves <em>frames</em> across a link</td>
</tr>
<tr>
<td>L1 Physical</td>
<td>moves <em>bits</em> across a physical medium</td>
</tr>
</tbody>
</table>
Each layer (except for L3) is implemented with different protocols

<table>
<thead>
<tr>
<th>layer</th>
<th>protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>L5</td>
<td>Application HTTP, SMTP, FTP, SIP, …</td>
</tr>
<tr>
<td>L4</td>
<td>Transport TCP, UDP, SCTP</td>
</tr>
<tr>
<td>L3</td>
<td>Network IP</td>
</tr>
<tr>
<td>L2</td>
<td>Link Ethernet, Wifi, (A/V)DSL, WiMAX, LTE, …</td>
</tr>
<tr>
<td>L1</td>
<td>Physical Twisted pair, fiber, coaxial cable, …</td>
</tr>
</tbody>
</table>
The Internet Protocol (IP) acts as a unifying, network, layer.
Each layer takes messages from the layer above, and *encapsulates* with its own header and/or trailer.
your laptop

Application: HTTP(S)
Transport: TCP/UDP
Network: IP
Link: Ethernet

Header:
Message: GET google.ch
In practice, layers are distributed on every network device.
Since when bits arrive they must make it to the application, all the layers exist on a host.
Routers act as **L3 gateway**
as such they implement L2 and L3

---

**Diagram Description:**

- **Application:** HTTP(S)
- **Transport:** TCP/UDP
- **Network:** IP
- **Link:** Ethernet

**Router Diagram:**

- The router acts as an **L3 gateway**.
- It implements both L2 and L3 functionalities.

---

**Network Structure:**

- **eth0**, **eth1**, **eth2**

---

**Legend:**

- **Router**
- **L3 gateway**
Switches act as L2 gateway as such they only implement L2
Let’s see how it looks like in practice on a host, using Wireshark

https://www.wireshark.org
Modularity, based on abstraction, is the way things get done.

— Barbara Liskov, MIT

But what is the right modularity?
What is the *right* modularity?

The need for stable interfaces

The end-to-end principle

The fate-sharing principle
Each layer is implemented with different protocols and technologies.
Network stack challenges at increasing speeds

The 100Gbit/s challenge

Jesper Dangaard Brouer
Red Hat Inc.

Linux Conf Au, New Zealand, January 2015
MICROSOFT SUPERCHARGES BING SEARCH WITH PROGRAMMABLE CHIPS

BEN BENDER

Burger works inside Microsoft Research—the group where the tech giant explores blue-sky ideas—and in November 2012, he pitched a radical new concept to Qi Lu, the man who oversees Microsoft’s Bing web search engine. He wanted to completely change the machines that make Bing run, arming them with a new kind of computer processor.
The interfaces between layers should be durable

Applications
...built on...

Reliable (or unreliable) transport
...built on...

Best-effort global packet delivery
...built on...

Best-effort local packet delivery
...built on...

Physical transfer of bits

Interfaces changing often implies broken layering
The end-end principle
What if no reliable transport is provided?

Every application that needs reliability has to engineer it from scratch

- Programmer burden
- Much higher likelihood of bugs
- Wasteful effort

But what if the network layer tried to provide reliable delivery?
What if the network layer tried to provide reliable delivery?

Reliable (or unreliable) transport
...built on...

Best-effort global packet delivery
Reliable

#1: My voice call wants speedy delivery, even if it’s lossy …
What if the network layer tried to provide reliable delivery?

Reliable (or unreliable) transport
...built on...

Best-effort global packet delivery
Reliable

#2: Can the network even achieve this at all?
Example: reliably transfer file from host A to B

**Solution 1:**

Check reliability at every step (involving network layer)

**Solution 2:**

Allow unreliable steps (network layer is best-effort)

B checks and tells A to retry on failure
Example: reliably transfer file from host A to B

Solution 1:

Check reliability at every step (involving network layer)

Problem: Bugs, failures are a truth of life
Example: reliably transfer file from host A to B

Solution 1:
Check reliability at every step (involving network layer)

Problem:
Bugs, failures are a truth of life

It’s not reliable transfer, if A-B depend on network that can fail
Example: reliably transfer file from host A to B

**Solution 2:**

Allow unreliable steps (network layer is best-effort).
B checks correctness. On failure, B tells A to retry.

*Can still fail, but only if A / B themselves fail.*

Solution 2 depends only on what end-points themselves control
Example: reliably transfer file from host A to B

“the end-to-end check of the file transfer application must still be implemented no matter how reliable the communication system becomes”
Question: should we ever implement reliability in the network?

A

P (error) = 0.1

B
Question: should we ever implement reliability in the network?

Yes, some, to reduce the number of end-end retries needed!

\[
P\text{(retry)} = 1 - 0.90^{10} = 0.65
\]

\[
P\text{(retry)} = 1 - 0.99^{10} = 0.10
\]
Implementing reliability in the network …

… does not reduce end-host complexity

… does increase network complexity

… often imposes overhead for apps that don’t need it

… but can enhance performance in some cases
Beyond a host-centric view …

Hosts are not the only stakeholders!

Networks want protection from malicious hosts

Things like firewalls are easier to implement in-network
The fate-sharing principle
A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.

— Leslie Lamport, Microsoft Research

How do we prevent this?
The fate-sharing principle

When storing state in a distributed system, co-locate it with entities that rely on that state.

State is lost only if those entities fail; then it doesn’t matter.

Example: network connection state at end hosts.
Food for thought ...

• When do we break layering?
• How do we define “end”?
  • TCP vs. application?
  • iPhone vs. iWatch?
  • Distributed applications?
• When do we break the end-to-end argument?
Computer Networks

Part 1: Overview

#1 What is a network made of?

#2 How is it shared?

#3 How does communication happen?

#4 How do we characterize it?
Performance = bandwidth?
To get many megabits-per-second ...
To get many megabits-per-second ...

4400 km
80 Km / hour
1 TB USB stick
= 40 Mbps

... but 55 hours
A network *connection* is characterized by its delay, loss rate, and throughput.

How long does it take for a packet to reach the destination?

What fraction of packets sent to a destination are dropped?

At what rate is the destination receiving data from the source?
A network *connection* is characterized by its delay, loss rate, and throughput.
Sources of network delays
Each packet suffers from several types of delays at each node along the path.

\[
\text{transmission delay} + \text{propagation delay} + \text{processing delay} + \text{queuing delay} = \text{total delay}
\]

- Transmission delay: due to link properties
- Propagation delay: due to traffic mix & switch internals
Overall, the main culprits for the overall delay are the transmission, propagation and queuing delays

\[
\text{transmission delay} + \text{propagation delay} + \text{processing delay} + \text{queuing delay} = \text{total delay}
\]

tend to be tiny
The transmission delay is the amount of time required to push all of the bits onto the link.

\[
\text{Transmission delay} = \frac{\text{packet size}}{\text{link bandwidth}} \quad [\text{sec}]
\]

Example:

\[
\frac{1000 \text{ bits}}{100 \text{ Mbps}} = 10 \mu\text{sec}
\]
The propagation delay is the amount of time required for a bit to travel to the end of the link.

\[
\text{Propagation delay} = \frac{\text{link length}}{\text{propagation speed}} \quad [\text{sec}]
\]

\[
\text{propagation speed} = \frac{\text{fraction of speed of light}}{[\text{m/sec}]}
\]

Example

\[
\frac{30000 \text{ m}}{2.10^8 \text{ m/sec}} = 150 \mu\text{sec}
\]

(speed of light in fiber)
How long does it take for a 100 Byte packet to travel from A to B? (not considering queuing for now)

Time to transmit one bit = $10^{-6}$s

Time to transmit 800 bits = $800 \times 10^{-6}$s

Time when that bit reaches B: $10^{-6} + 10^{-3}$s

The last bit reaches B at $(800 \times 10^{-6}) + 10^{-3}$s = 1.8ms
If we have a 1 Gbps link, the total time decreases to 1.08ms

Time to transmit one bit = $10^{-9}$s

Time to transmit 800 bits = $800 \times 10^{-9}$s

Time when that bit reaches B: $10^{-9} + 10^{-3}$s

The last bit reaches B at $(800 \times 10^{-9}) + 10^{-3}$s = 1.08ms
If we now exchange a 1GB file split in 100B packets

The last bit reaches B at

\[
(10^7 \times 800 \times 10^{-9}) + 10^{-3}s = 8001ms
\]
Different transmission characteristics imply different tradeoffs in terms of which delay dominates.

<table>
<thead>
<tr>
<th>Data Size</th>
<th>Payload</th>
<th>Link Speed</th>
<th>Delay Dominance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^7 \times 100$B</td>
<td>pkt</td>
<td>1Gbps link</td>
<td>transmission delay dominates</td>
</tr>
<tr>
<td>$1 \times 100$B</td>
<td>pkt</td>
<td>1Gbps link</td>
<td>propagation delay dominates</td>
</tr>
<tr>
<td>$1 \times 100$B</td>
<td>pkt</td>
<td>1Mbps link</td>
<td>both matter</td>
</tr>
</tbody>
</table>

Often, we can’t know in advance which one matters!
The queuing delay is the amount of time a packet waits (in a buffer) to be transmitted on a link.

Queuing delay is the hardest to evaluate as it varies from packet to packet.

It is characterized with statistical measures e.g., average delay & variance, probability of exceeding x.
Queuing delay depends on the traffic pattern
No overload
Queuing delay depends on the traffic pattern
Queues absorb transient bursts, but introduce queueing delays.
The time a packet has to sit in a buffer before being processed depends on the traffic pattern

- arrival rate at the queue
- transmission rate of the outgoing link
- traffic burstiness
average packet arrival rate $a$ [packet/sec]

transmission rate of outgoing link $R$ [bit/sec]

fixed packet length $L$ [bit]

average bits arrival rate $La$ [bit/sec]

traffic intensity $La/R$
When the **traffic intensity** is >1, the queue will increase without bound, and so does the queuing delay.

**Golden rule**

Design your queuing system, so that it operates far from that point.
When the traffic intensity is $\leq 1$, queueing delay depends on the burst size.
Why does delay matter?
Can you perceive…

1 second?

500 milliseconds?

100 milliseconds?

10 milliseconds?

spoken syllable ~150 - 200 ms
Can you perceive…

1 second?

500 milliseconds?

100 milliseconds?

10 milliseconds?

spoken syllable

Hiromi Uehara

“Kung Fu World Champion”
Can you perceive…

1 second?

500 milliseconds?

100 milliseconds?

10 milliseconds?

← spoken syllable
~150 - 200 ms

← Hiromi Uehara
88 ms per note
We care about very small delays!

In online services

<table>
<thead>
<tr>
<th></th>
<th>Delay</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>+100ms</td>
<td>-1% revenue</td>
</tr>
<tr>
<td>Bing</td>
<td>+500ms</td>
<td>-1.2% revenue</td>
</tr>
<tr>
<td>Google</td>
<td>+400ms</td>
<td>-0.7% searches</td>
</tr>
</tbody>
</table>

In gaming

What is a nanosecond, a microsecond, … ?

Via Colin Scott (https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html)
What is a nanosecond, a microsecond, … ?

“Amazing Grace” Hopper
What is a nanosecond, 
a microsecond, … ?
What are Internet delays like?
What do Internet delays look like in practice?

[~]$ ping google.ch
What do Internet delays look like in practice?

```bash
[~]$ ping google.ch
PING google.ch (172.217.16.131): 56 data bytes
64 bytes from 172.217.16.131: icmp_seq=0 ttl=56 time=6.428 ms
64 bytes from 172.217.16.131: icmp_seq=1 ttl=56 time=1.215 ms
64 bytes from 172.217.16.131: icmp_seq=2 ttl=56 time=1.200 ms
64 bytes from 172.217.16.131: icmp_seq=3 ttl=56 time=1.180 ms
64 bytes from 172.217.16.131: icmp_seq=4 ttl=56 time=1.195 ms
64 bytes from 172.217.16.131: icmp_seq=5 ttl=56 time=1.144 ms
64 bytes from 172.217.16.131: icmp_seq=6 ttl=56 time=1.156 ms
64 bytes from 172.217.16.131: icmp_seq=7 ttl=56 time=1.096 ms
64 bytes from 172.217.16.131: icmp_seq=8 ttl=56 time=1.175 ms
64 bytes from 172.217.16.131: icmp_seq=9 ttl=56 time=1.181 ms
64 bytes from 172.217.16.131: icmp_seq=10 ttl=56 time=1.138 ms
64 bytes from 172.217.16.131: icmp_seq=11 ttl=56 time=1.130 ms
64 bytes from 172.217.16.131: icmp_seq=12 ttl=56 time=1.227 ms
64 bytes from 172.217.16.131: icmp_seq=13 ttl=56 time=1.162 ms
^C
--- google.ch ping statistics ---
14 packets transmitted, 14 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 1.096/1.545/6.428/1.355 ms
```
we describe QUIC’s impact on Google Search and YouTube. These Amazon estimates that every 100 ms increase in latency cuts profits well under 20% of Search Latency and Video Latency. An almost whether a transport change is working correctly, but application and trends are similar.

9 top achieve a 0-RTT handshake, which is at least a 2-RTT latency the 0-RTT handshake: about 88% of QUIC connections from desk-handshake improvements, most of the latency reduction comes from of all connections have a minimum RTT larger than 300ms. Of the that users with high RTTs are in a significant tail: more than 20% of is higher, leading to larger gains in QUIC

The user's RTT increases, the impact of saving handshake round trips in Figure 9 which shows desktop latency reduction for users in Search Latency. The percentile data shows that QUIC improvements for these highly optimized applications have direct revenue performance for these highly optimized Web applications, and consequently

First, networking remains just one constituent of end-to-end ap-

Microbenchmarks such as the ones above are a useful measure of user-relevant metrics are a measure of the usefulness of the change. As shown in Table 1, users in QUIC

What do Internet delays look like in practice?

36% of connections over 100ms

[The QUIC Transport Protocol: Design and Internet-Scale Deployment, Langley et. al, SIGCOMM 2017]
Caveat: There are some obvious geolocation errors
A network connection is characterized by its delay, loss rate, and throughput.
In practice, queues are not infinite. There is an upper bound on queuing delay. 

queue

N+1 packets

packet size $L$ transmission rate: $R$

queuing delay upper bound: $N*\frac{L}{R}$
If the queue is persistently overloaded, it will eventually drop packets (loss)
Why does loss matter?

Sometimes, it doesn’t!
e.g., losing a negligible part of conversation

Sometimes, translates into service delays
“This packet didn’t arrive? I’ll try again.”

Can also send redundant information in advance
A, B, A \oplus B — incurs a capacity penalty
What does loss on the Internet look like?

![Graph showing average loss percentage (retransmissions) vs. minimum RTT to Google servers (milliseconds).]

- Avg. loss %
- (retransmissions)

Minimum RTT to Google servers (milliseconds)

[The QUIC Transport Protocol: Design and Internet-Scale Deployment, Langley et. al, SIGCOMM 2017]
A network *connection* is characterized by its delay, loss rate, and throughput.
The throughput is the rate at which a host receives data

Average throughput \( \text{[bits/sec]} \) = \( \frac{\text{data size [bits]}}{\text{transfer time [sec]}} \)
To compute throughput, one has to consider the bottleneck link.

Average throughput $\min(R_S, R_L) = \text{transmission rate of the bottleneck link}$
To compute throughput, one has to consider the bottleneck link and the intervening traffic.

\[
\text{if } 4 \times \min(R_S, R_L) > R \quad \text{the bottleneck is now in the core, providing each download } R/4 \text{ of throughput}
\]
Why does throughput matter?

Some applications need little bandwidth
- e.g., gaming typically needs few kbps

Some need a lot more
- e.g., HD video streaming often needs 5+ Mbps
What does throughput on the Internet look like?

Your Internet speed is 32 Mbps

Compare on speedtest.net

[fast.com]
What does throughput on the Internet look like?

[Map showing average Mbps across the world]

[Source: Akamai’s State of the Internet, 2017]
What does throughput on the Internet look like?

[Map showing the percentage of countries with 10+ Mbps Internet speed, with a color scale indicating the percentage range.]

[Akamai's State of the Internet, 2017]
What does throughput on the Internet look like?
A network *connection* is characterized by its delay, loss rate, and throughput.
As technology improves, throughput increases & delay decreases, except for propagation

(speed of light)

Data in fiber moves at ≈ 2c / 3

Significant room for improvement here!
Because of propagation delays, Content Delivery Networks move content closer to you

http://wwwnui.akamai.com/gnet/globe/index.html
Readings for this week

Kurose-Ross (KR) sections

1.5 Protocol layers and their service models

END-TO-END ARGUMENTS IN SYSTEM DESIGN
J.H. Saltzer, D.P. Reed and D.D. Clark*
M.I.T. Laboratory for Computer Science