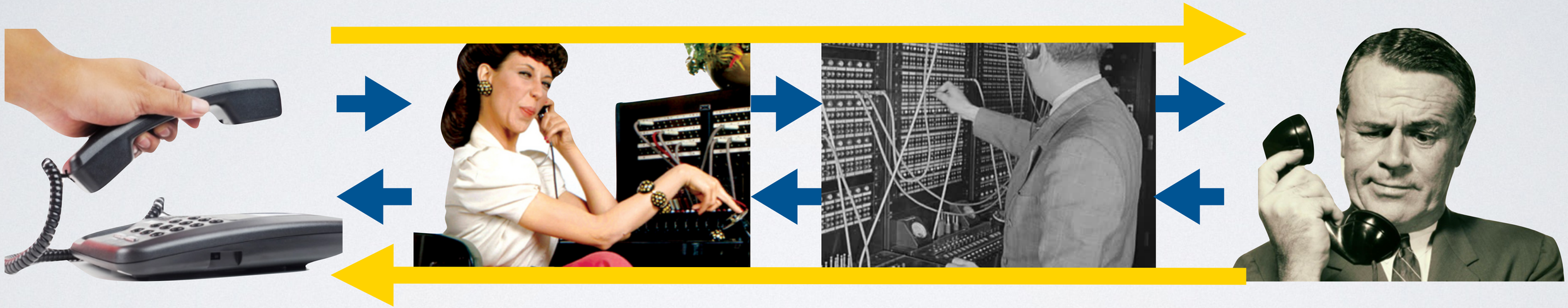
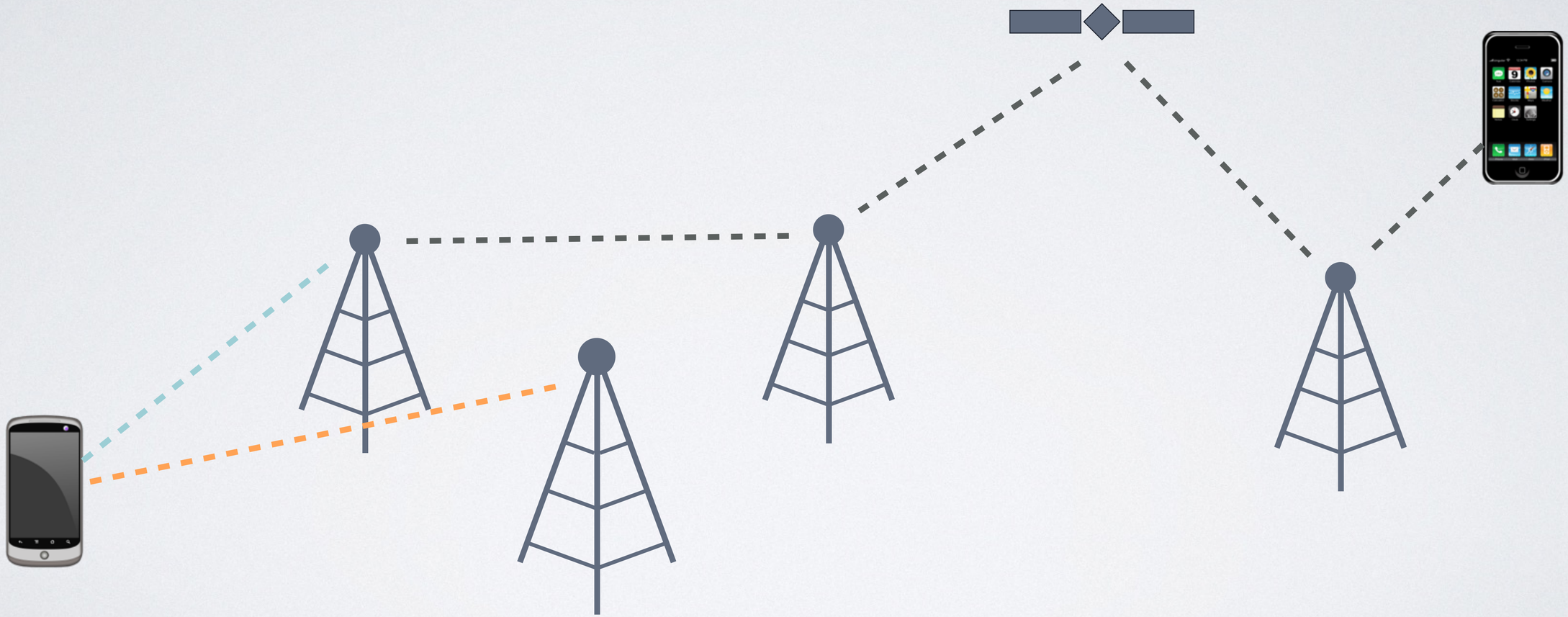
Internetworking

Circuit Switching	Packet Switching
Original phone system had a purely circuit switched foundation. All information within a call takes exactly the same fixed path. Routing only occurs before the call	The Internet protocol (IP) provides a decentralized packet-switched foundation. Each packet is individually and continuously routed, based on its destination address

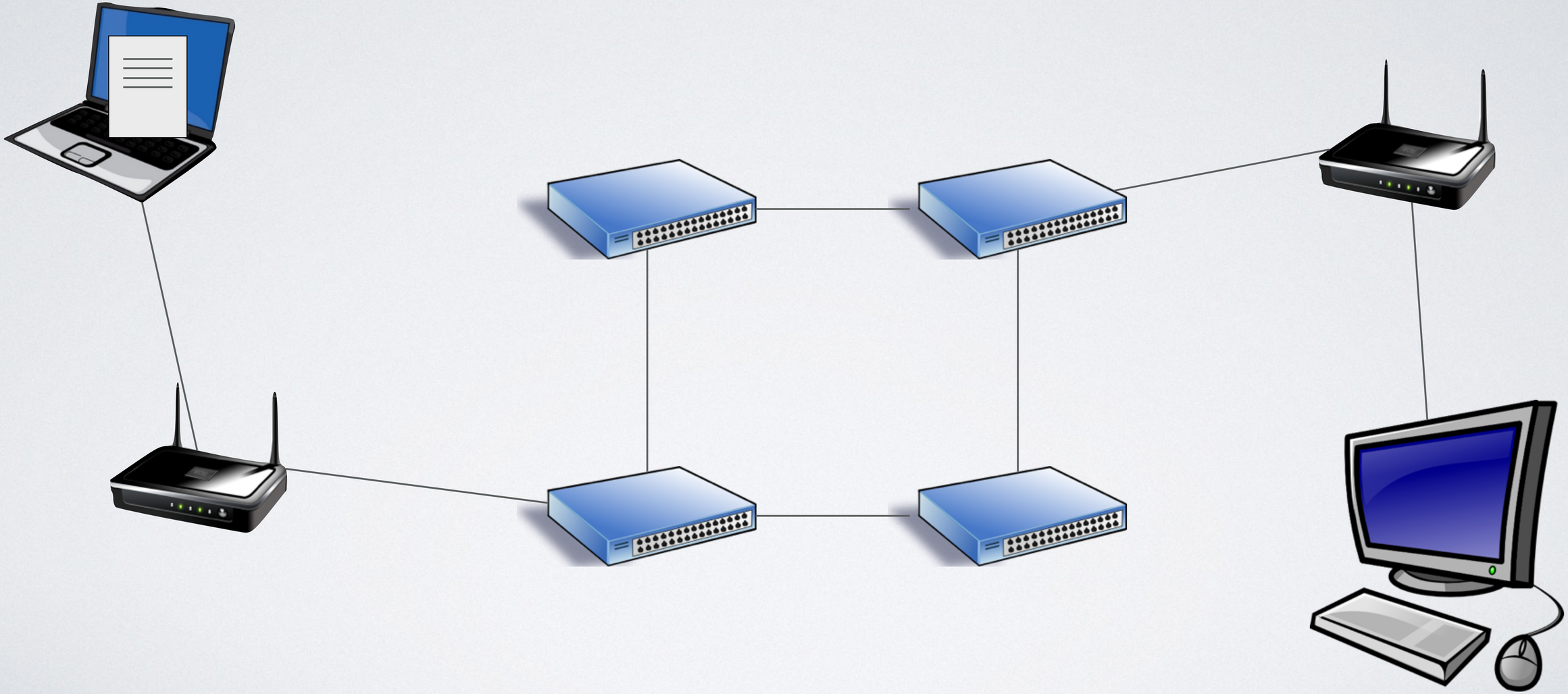
Circuit-Switching



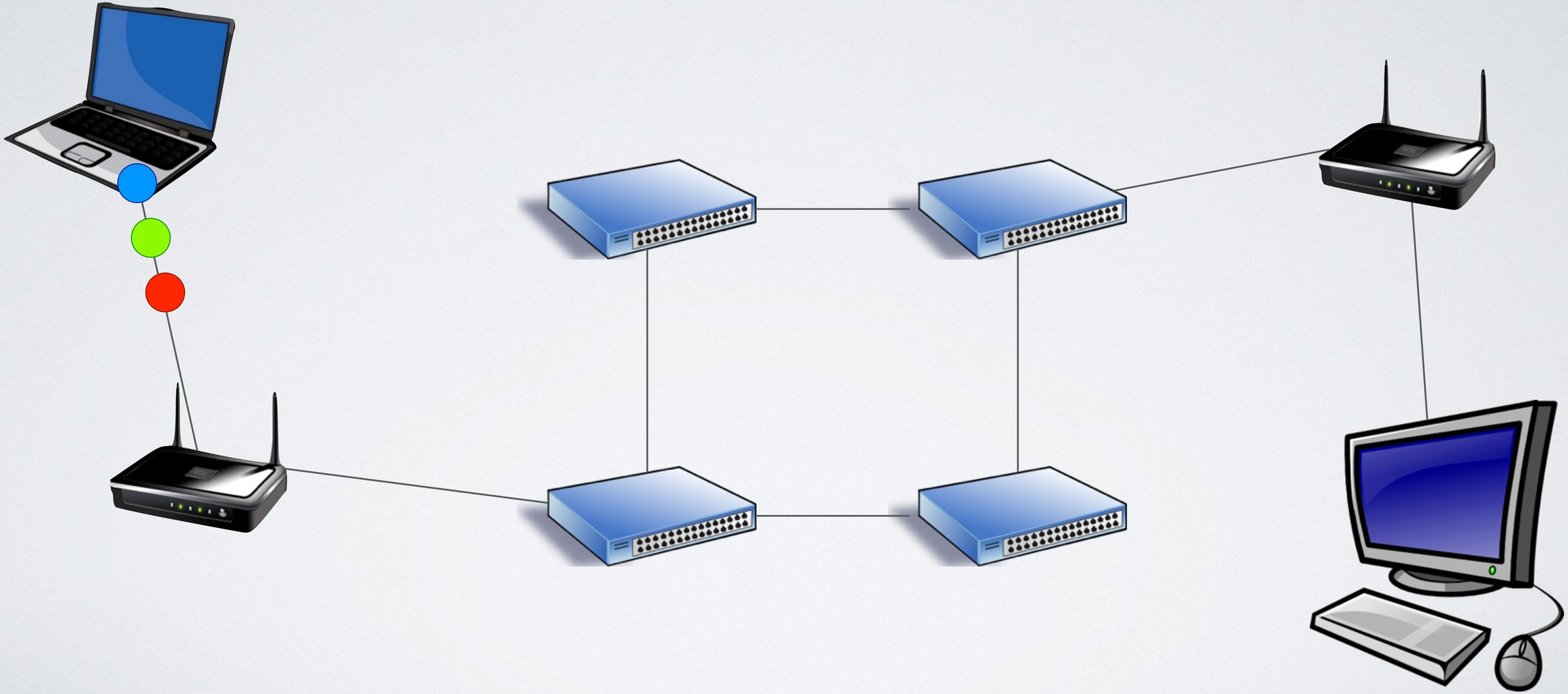
Circuit-Switching



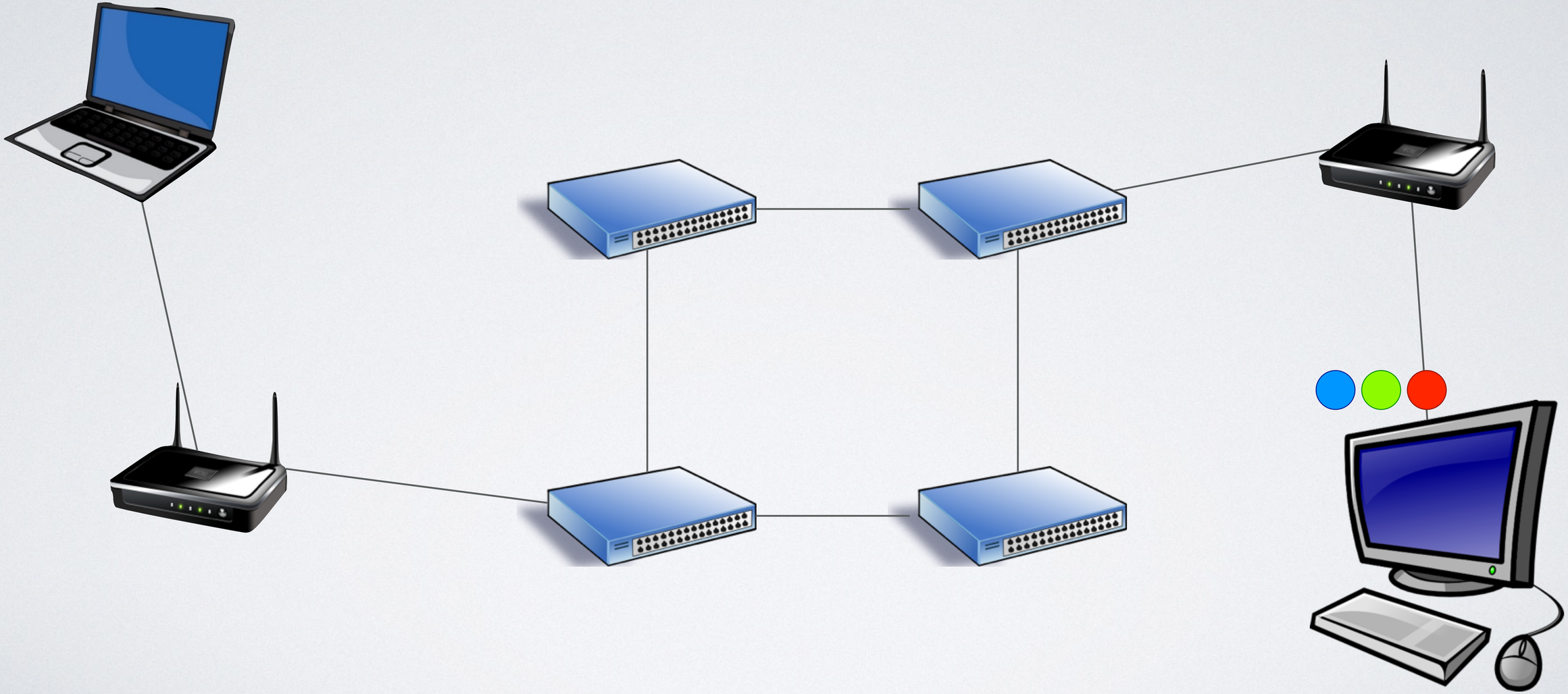
Packet-Switching



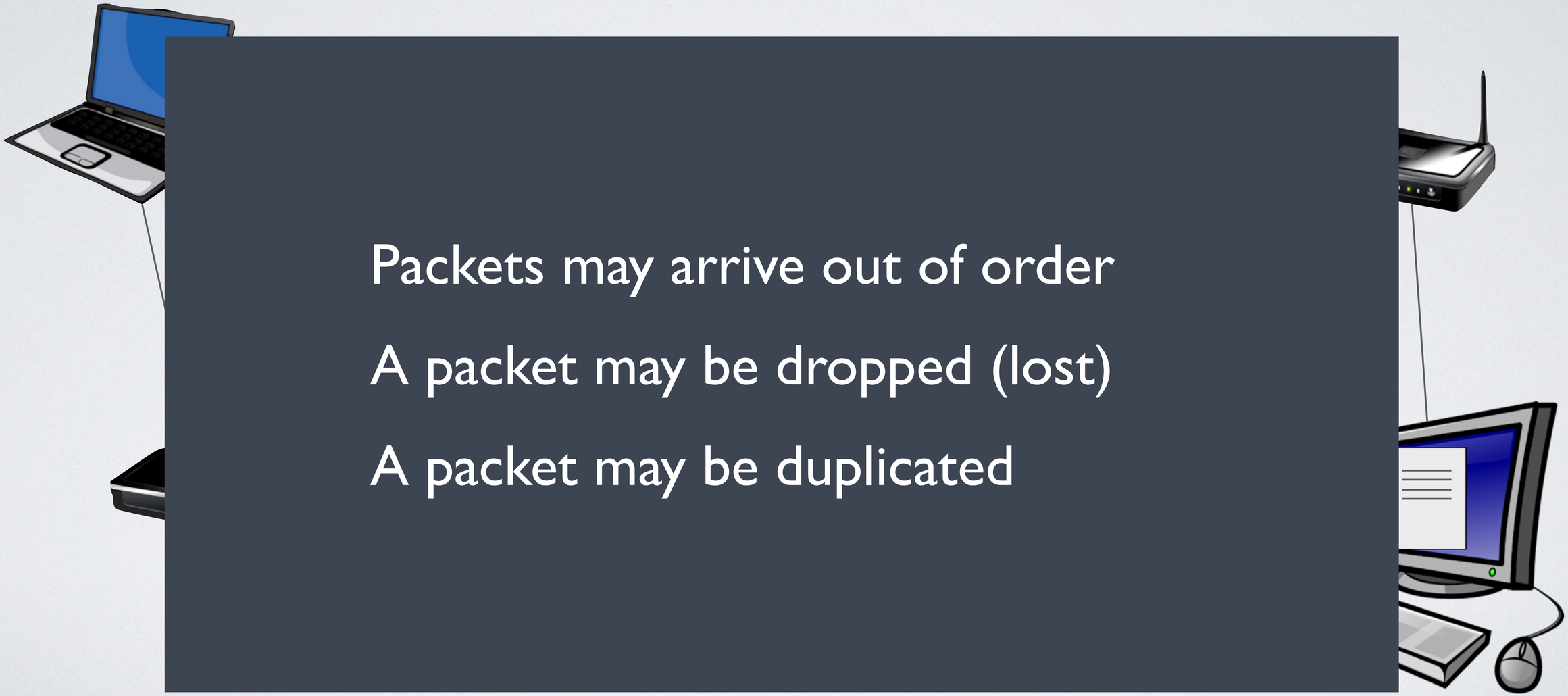
Packet-Switching



Packet-Switching



Packet-Switching



Packets may arrive out of order

A packet may be dropped (lost)

A packet may be duplicated

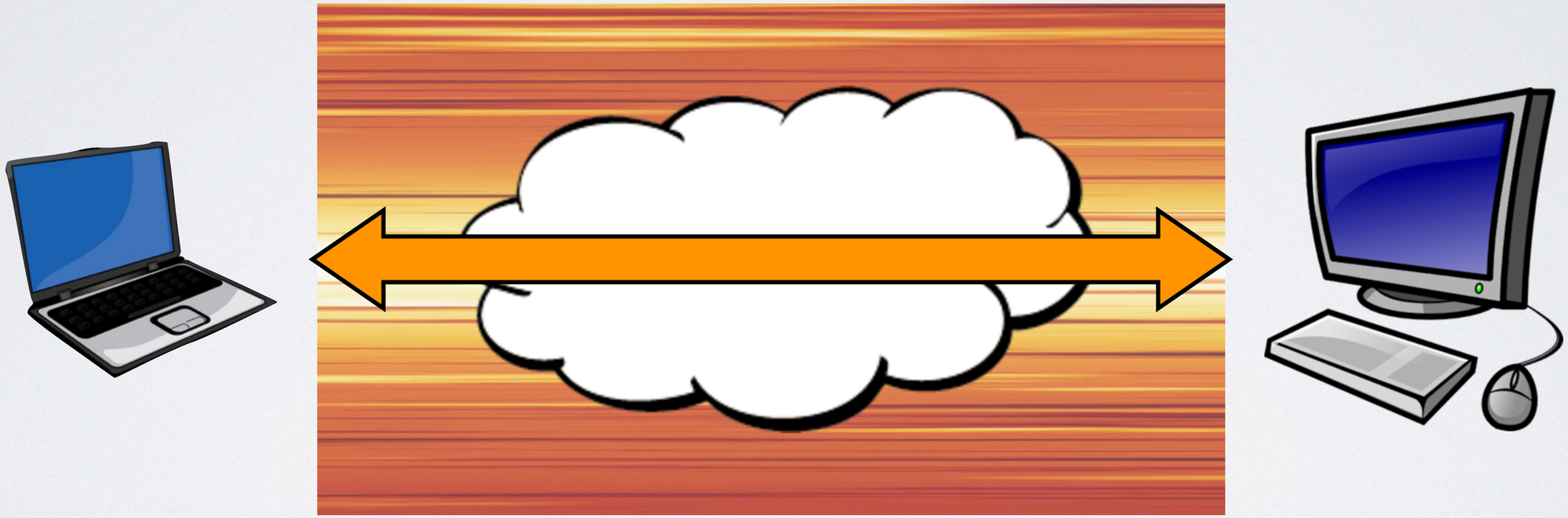
TCP (Transmission Control Protocol)

Connection-Oriented Protocol

Favors Reliability over Speed

UDP (Universal Datagram Protocol)

Connectionless Protocol



Favors Speed over Reliability

	TCP	UDP
Connection	Connection Oriented Handshake before user data	Connectionless No handshake
Reliability	Data remains intact and arrives in same order it was sent	No guarantee that messages or packets sent will reach at all
Speed	Slower than UDP	Faster because no error checking
Weight	Heavyweight – more overhead	Lightweight – less overhead
Usage	Useful for documents and downloads	Useful for live media and small queries
Popularity	Most common internet protocol	Growing in popularity

Pause and Think

Which applications use TCP, and which use UDP?

TCP	Amazon
TCP	Dropbox
UDP	DNS Lookup
TCP	Facebook
TCP	Gmail
UDP	VOIP
UDP	World of Warcraft
TCP?	Netflix

OSI Reference Model

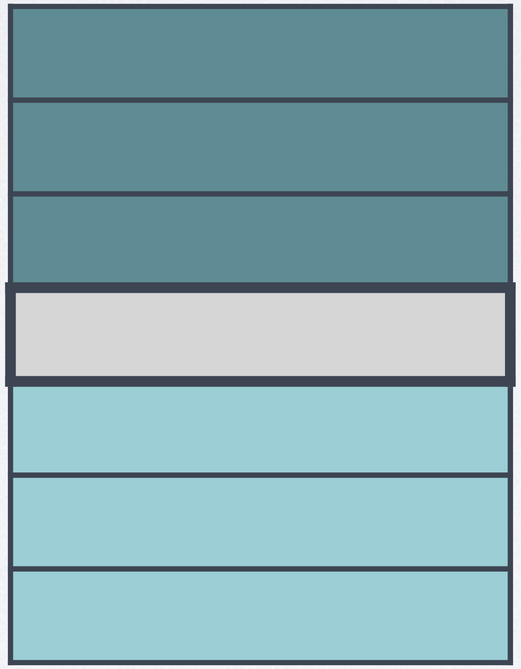
Protocol Reference Models

provide an abstract view of the network, and
group functionality into layers

Each layer:

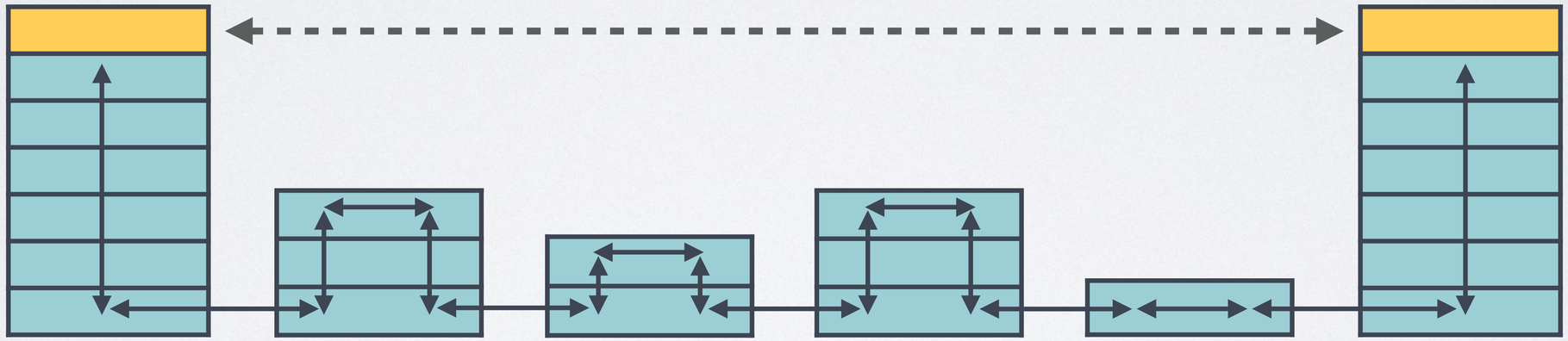
Provides services to the upper layers

Uses the services of the lower layers



Layers have one or more alternate protocols

Peer-to-Peer Communication



OSI Reference Model

7	**Application Layer**	Ensure communication between distributed software components
6	**Presentation Layer**	Convert operating system standards to or from network representations
5	**Session Layer**	Manages connections between local and remote application
4	**Transport Layer**	Deliver data from operating system to operating system
3	**Network Layer**	Deliver packets with routing services across multiple data links
2	**Data Link Layer**	Groups bits into addressed frames for transmission over physical links
1	**Physical Layer**	Deliver bits over physical link (or wireless)

OSI Reference Model

7	Application Layer	
6	Presentation Layer	
5	Session Layer	
4	Transport Layer	
3	Network Layer	
2	Data Link Layer	
1	Physical Layer	

OSI Reference Model (by example)

7	**Application Layer**	HTTP, FTP, SMTP, and a whole lot more...
6	**Presentation Layer**	Secure Socket Layer (SSL) provides encryption
5	**Session Layer**	Various "Sockets" implementations
4	**Transport Layer**	TCP (connection-oriented), UDP (connectionless)
3	**Network Layer**	Internet Protocol (IP)
2	**Data Link Layer**	Ethernet, Token Ring, FDDI, ATM, and others...
1	**Physical Layer**	10Base-T, 100Base-TX, V.90, IEEE 802.11, and many more...

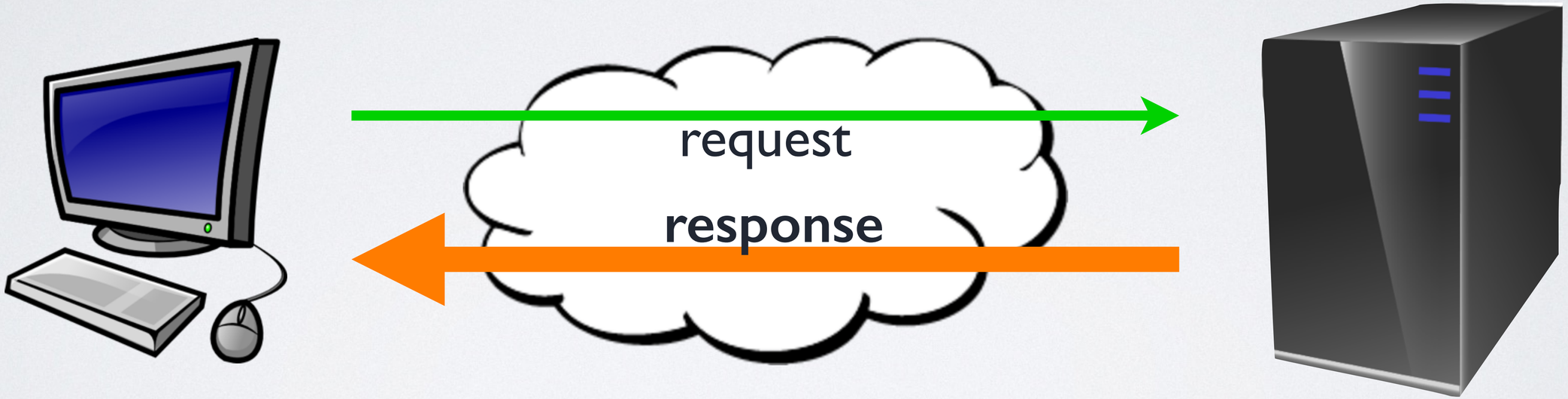
OSI Reference Model (by example)

7	**Application Layer**	HTTP, FTP, SMTP, and a whole lot more...
6	**Presentation Layer**	Secure Socket Layer (SSL) provides encryption
5	**Session Layer**	Various "Sockets" implementations
4	**Transport Layer**	TCP (connection-oriented), UDP (connectionless)
3	**Network Layer**	Internet Protocol (IP)
2	**Data Link Layer**	Ethernet, Token Ring, FDDI, ATM, and others...
1	**Physical Layer**	10Base-T, 100Base-TX, V.90, IEEE 802.11, and many more...

Distributed Computing Models

HTTP (Web)
SMTP (Email)
FTP (download)

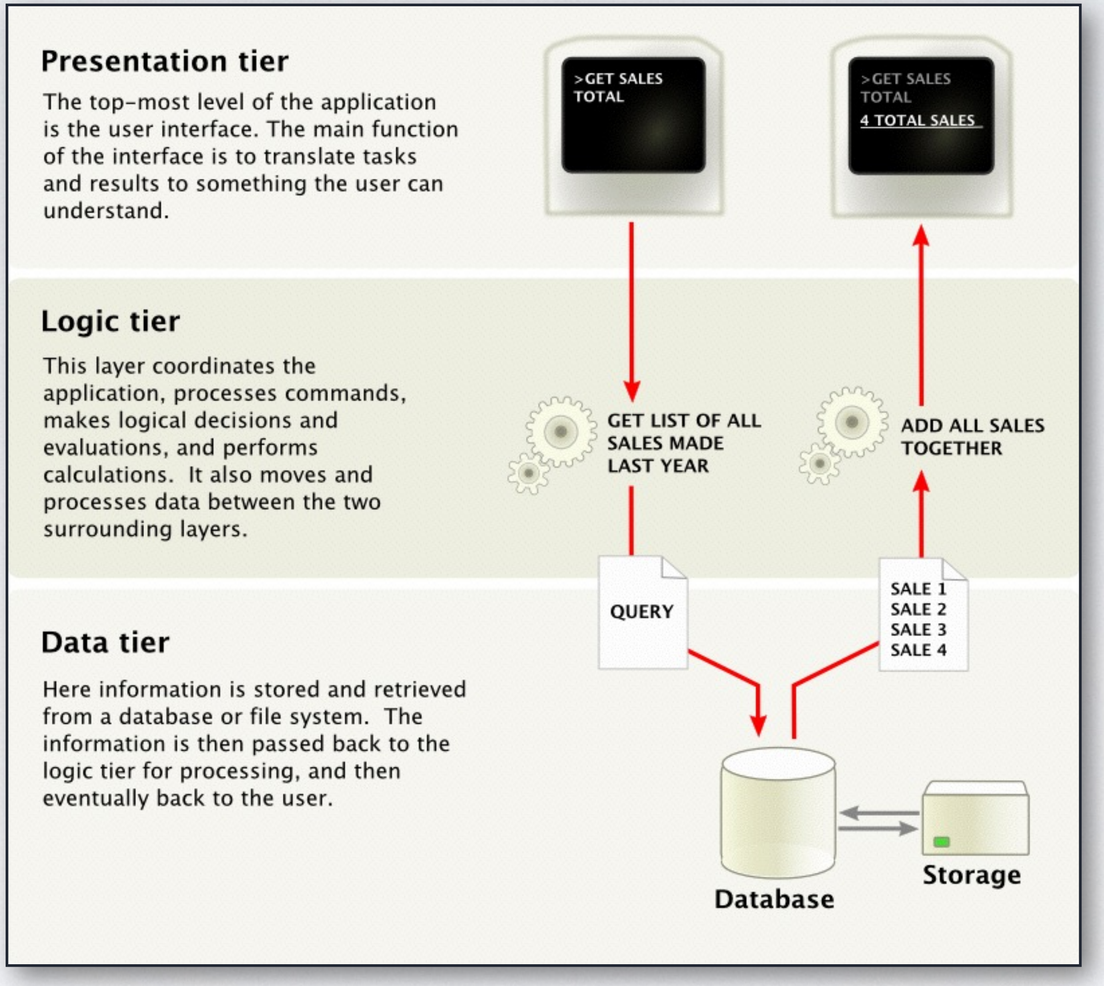
Client-Server Model



Three Tier Model

The logic tier can become complex and split into multiple tiers

When this occurs, it's called a **multi-tier** model

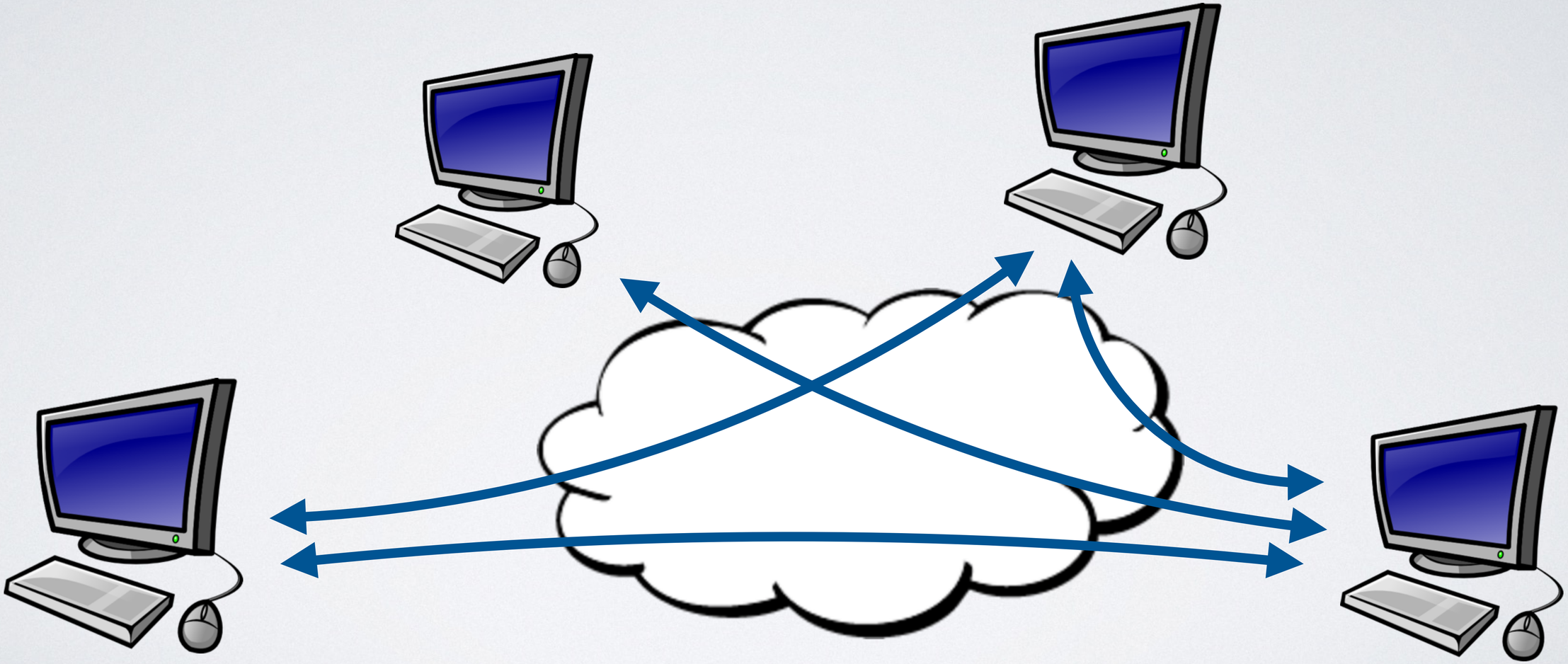


View

Controller

Model

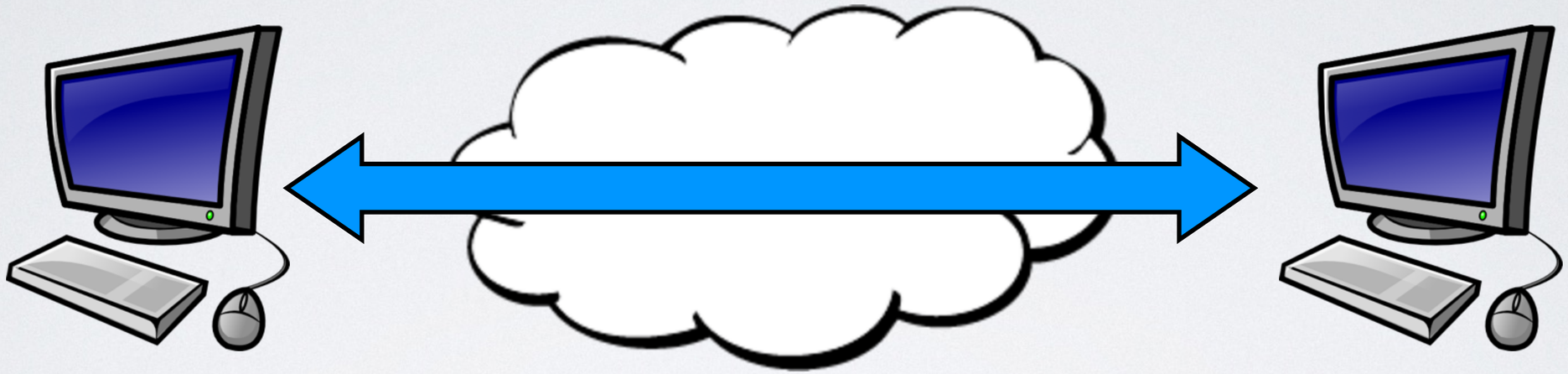
Peer-to-Peer Model



Sockets and Ports

Sockets and Ports

A network socket is an **endpoint** of a connection across a computer network.



```
Socket socket = getSocket(type = "TCP")
connect(socket, address = "8.8.8.8", port = "80")
send(socket, "Hello, world!")
close(socket)
```

IPv4, IPv6, and DNS

Google Public DNS IP addresses

The Google Public DNS IP addresses (IPv4) are as follows:

8.8.8.8
8.8.4.4

The Google Public DNS IPv6 addresses are as follows:

2001:4860:4860::8888
2001:4860:4860::8844

You can use either address as your primary or secondary DNS server. You can specify both addresses, but do not specify one address as both primary and secondary.

You can configure Google Public DNS addresses for either IPv4 or IPv6 connections, or both.