COMP 307

Principles of Web Development

Unit 1 – The Internet Landscape

Lecture 2:

About Networks

Lecture Outline

- Network architecture
- Packets & Why ASCII?
- Network Protocols
- Wireshark

Readings

- Internet and World Wide Web PDF
  - 1.5 to 1.9
  - Other links about the Internet:
    - http://netforbeginners.about.com
    - https://www.w3.org/
- What is a packet?
  - https://www.techtarget.com/searchnetworking/definition/packet Videos and reading
- Wireshark master class
  - https://www.youtube.com/watch?v=OU-A2EmVrKQ&t=3s
  - (https://www.wireshark.org/#learnWS Learn Wireshark

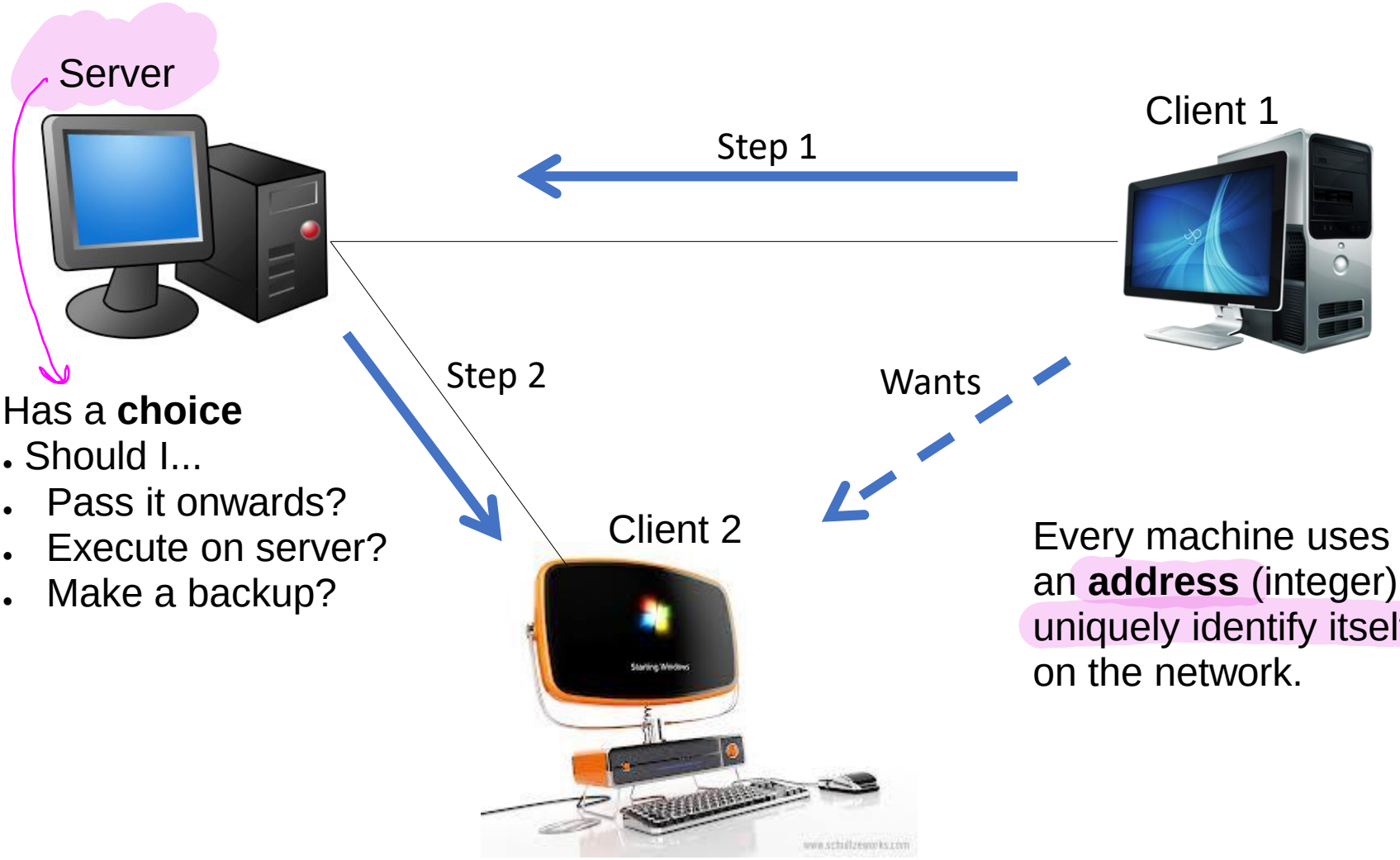
How do networks work?

Network

- Hardware
  - Computers, network cards, routers/hubs
- Medium:
  - A technology that interconnects machines: wires, radio (Bluetooth, narrow-band, Wi-Fi), Optical cables, lasers.
- Information:
  - a data structure passed between machines using the medium.
- Protocol:
  - rules for how to share data.

Client-Server Network

If Client 1 sends a message to Client 2, the message must pass through the server. There are no other wires.



Has a **choice**

- Should I...
- Pass it onwards?
- Execute on server?
- Make a backup?

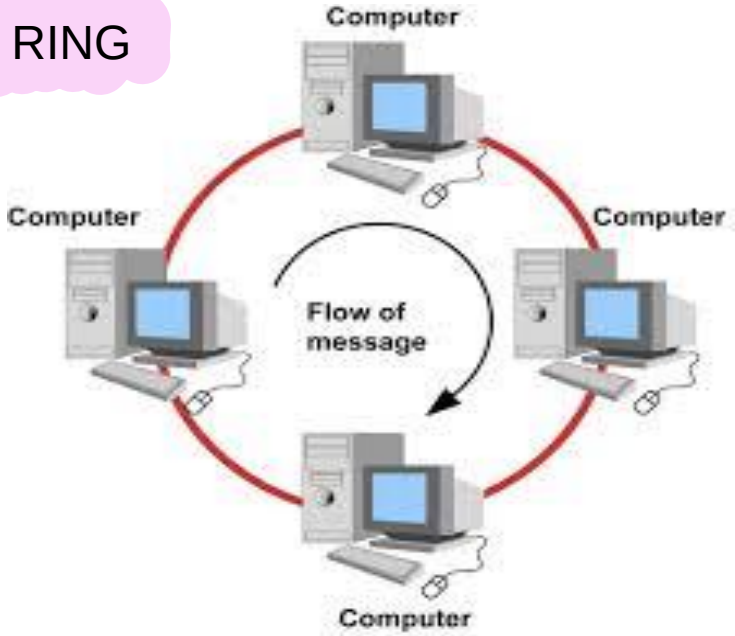
Every machine uses an **address** (integer) to uniquely identify itself on the network.

What kind of **privacy** do we have when A messages B? **Weak** points?

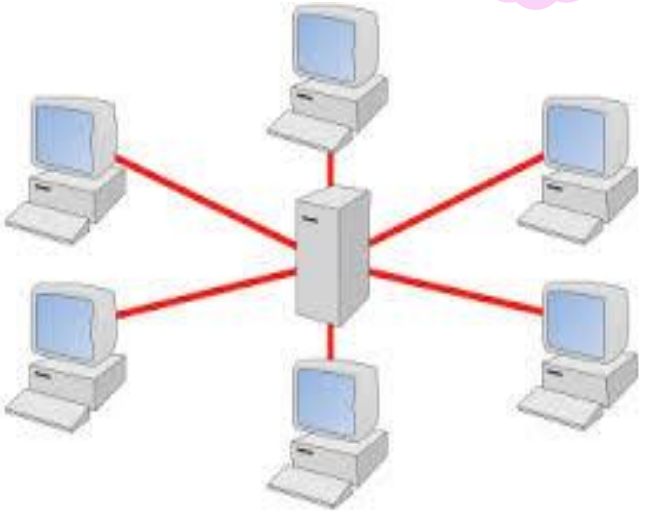
Topology

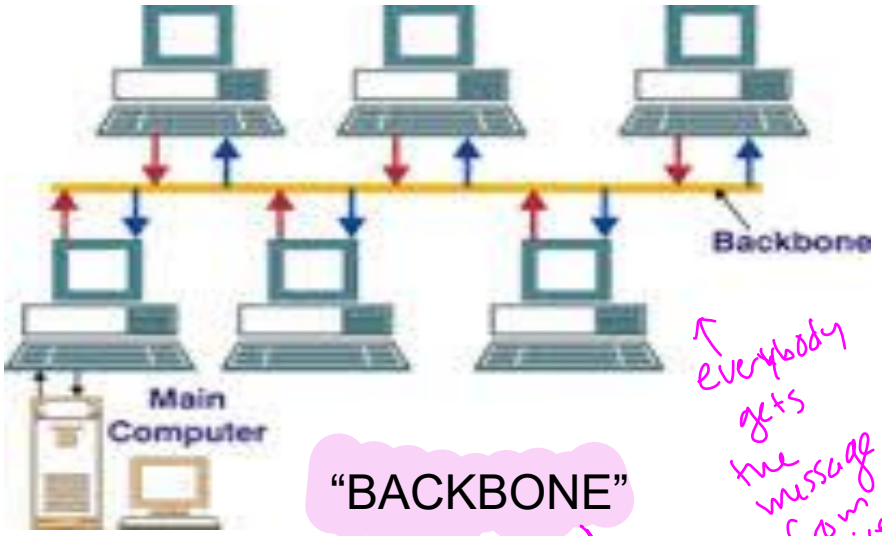
types of client-server networks

RING



STAR





"BACKBONE"

Internet based on this

everybody gets the message from wire → always a broadcast

TREE

Contents

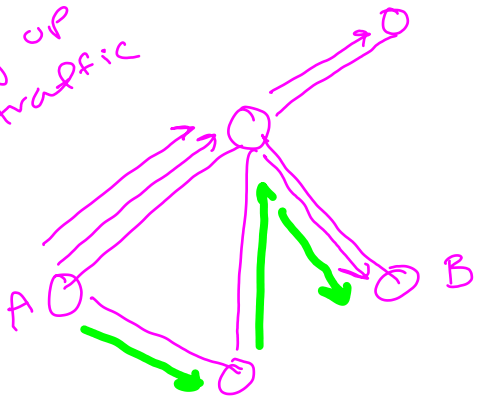
ISP

(Internet Service Provider)

- A special server that
  - Has members (free or $$)
  - Provides URL resolution to IP address
  - Has routing tables (map) or not
- Calculates shortest path
  - Can also broadcast
- Other features
  - Can do simple load balancing
  - Can do simple traffic avoidance

map of the internet is

splitting up the traffic

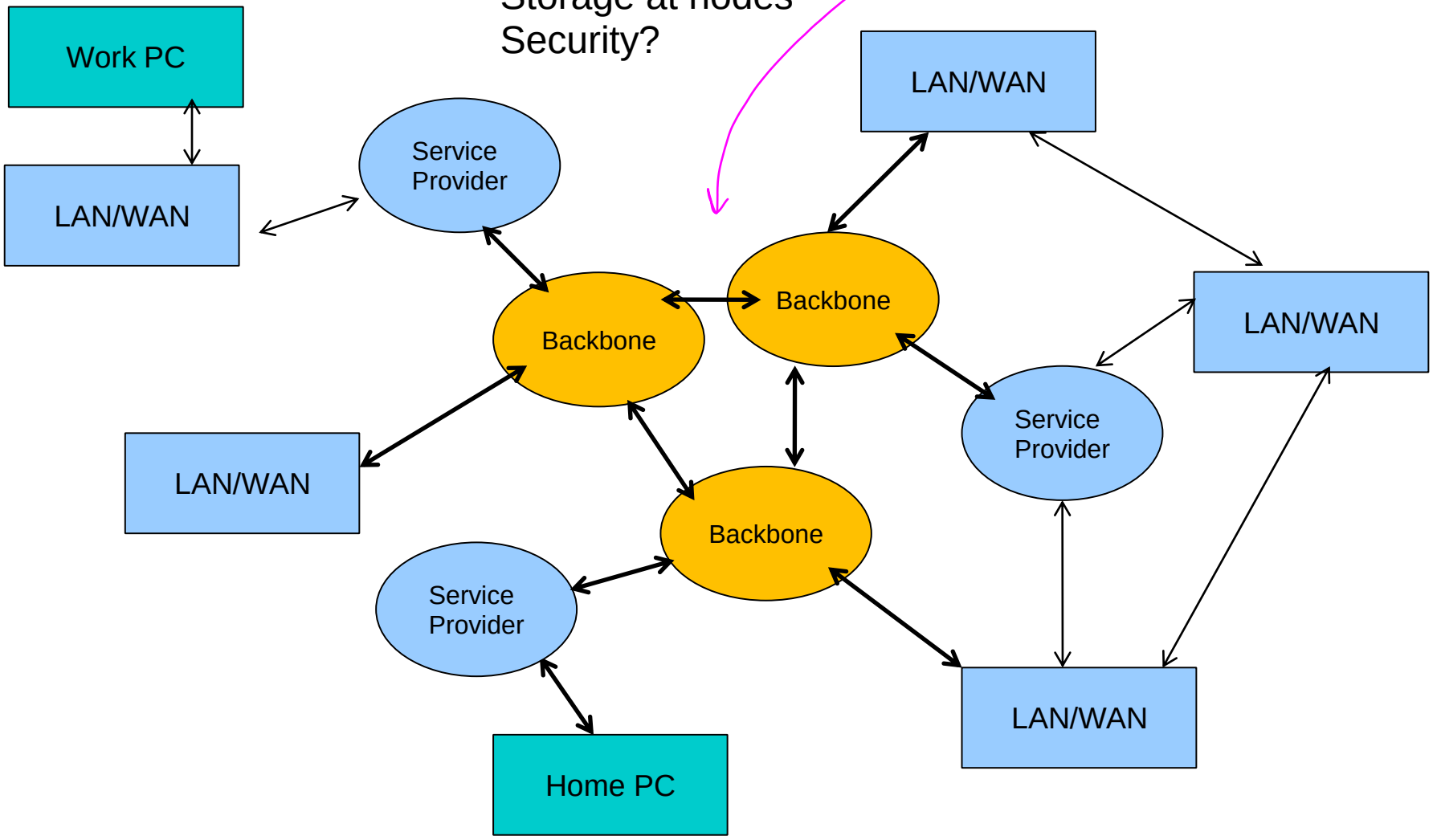


traffic on wires + servers

Internet Topology

backbone = hidden network interconnected

Storage at nodes
Security?

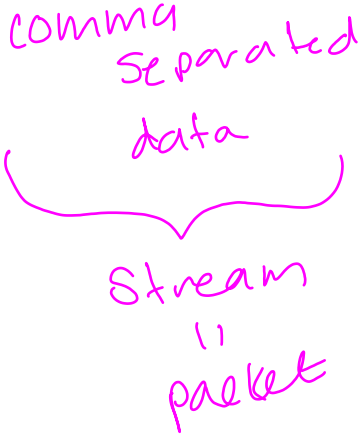


Internet Backbone is peer-to-peer

Very high-speed optical cable

What is the data structure?

Packets



ASCII with a format

(like comma-separated or CR/LF separated, or byte length separated)

Why do we use ASCII?

Because it is the most compatible method of sending information.

Integer, on some computers are 16-bit, 32-bit or 64-bit.

But ASCII is always 8-bit.

We need conversion functions…

bad for security
↓
only need text editor to read a packet

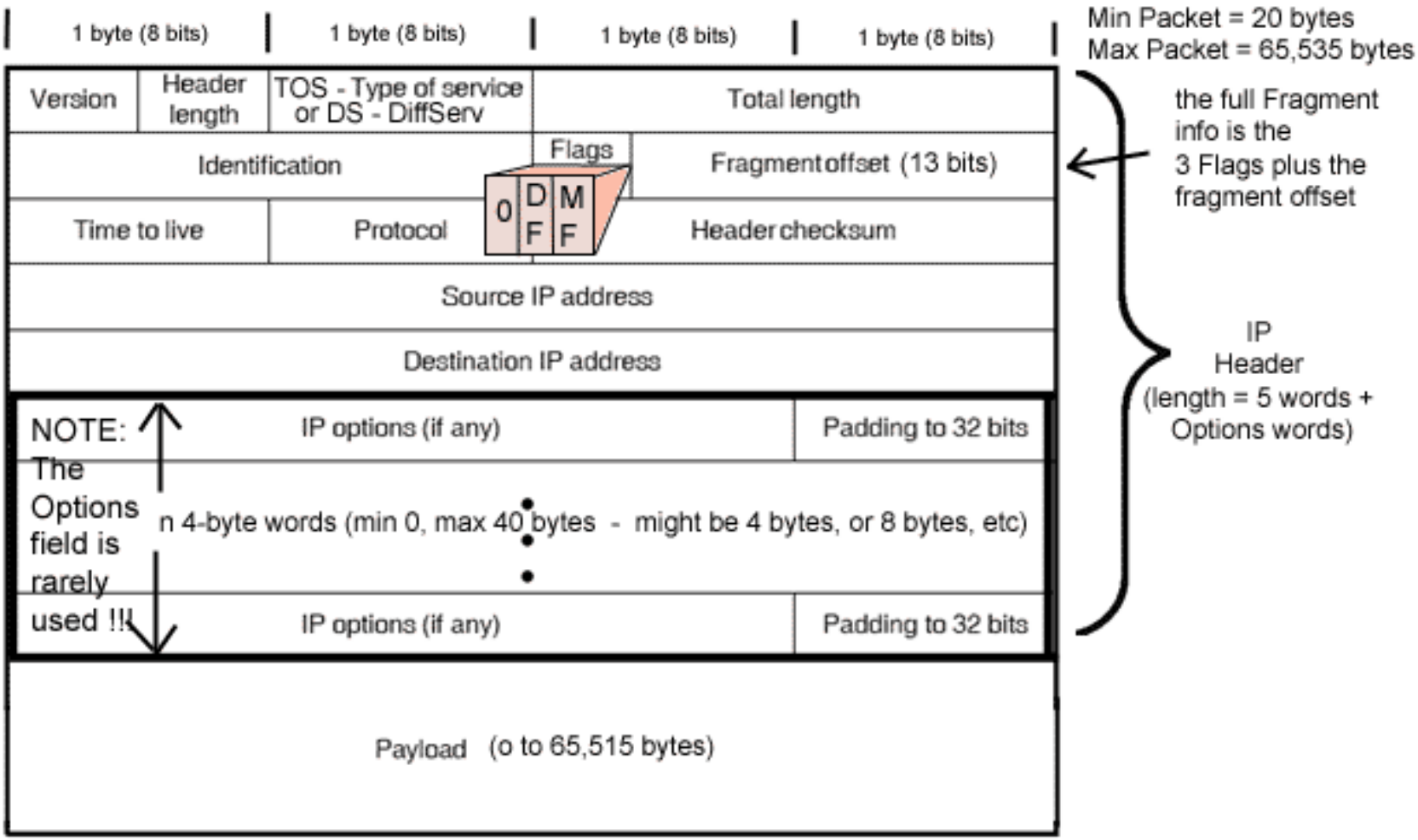
Simple Packet

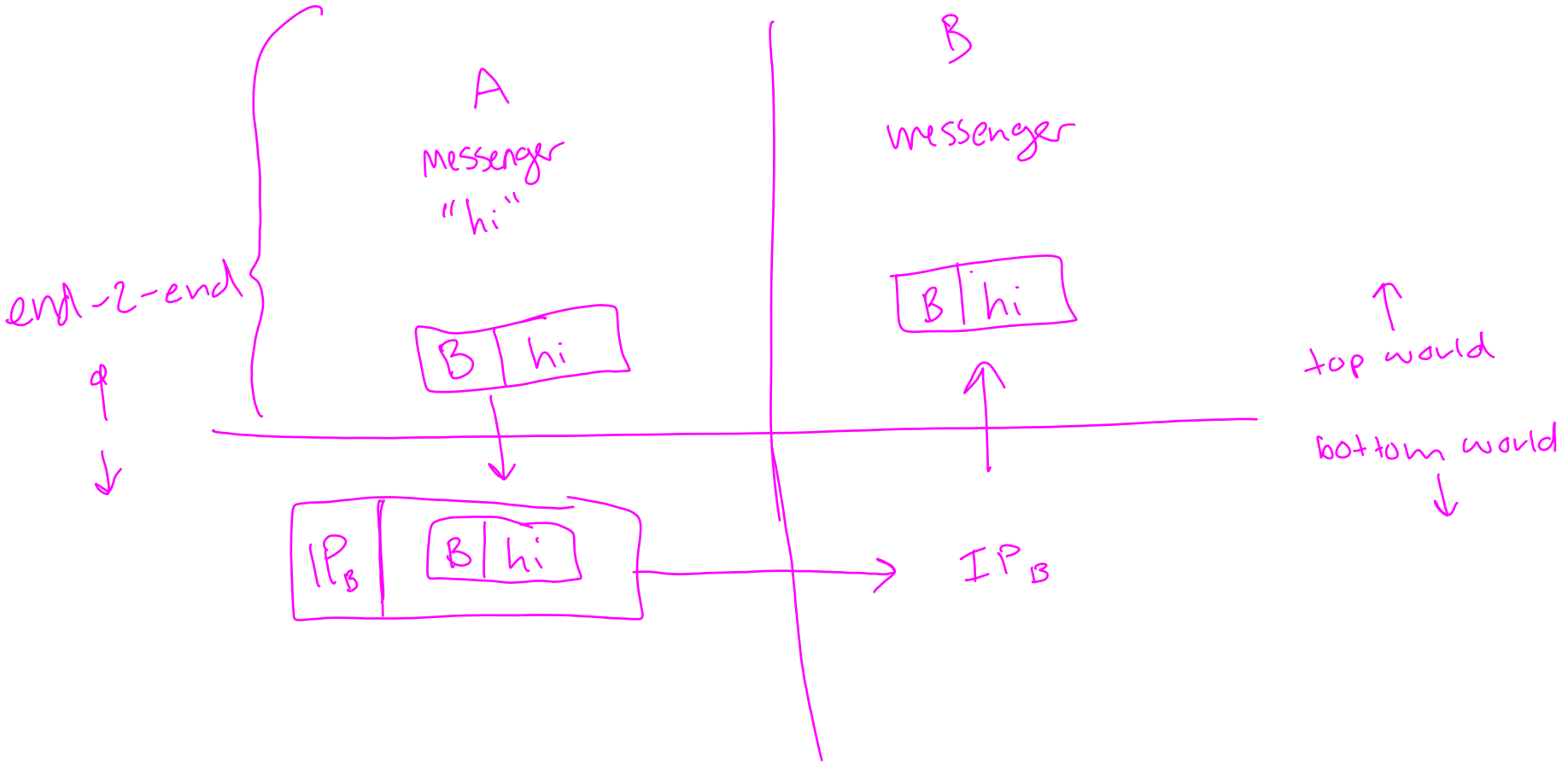
A data structure used to store and transmit messages between two locations.

Packet version #	Length in bytes
Source address	Destination address
Sequence number	Protocol
Message	Error correcting bits

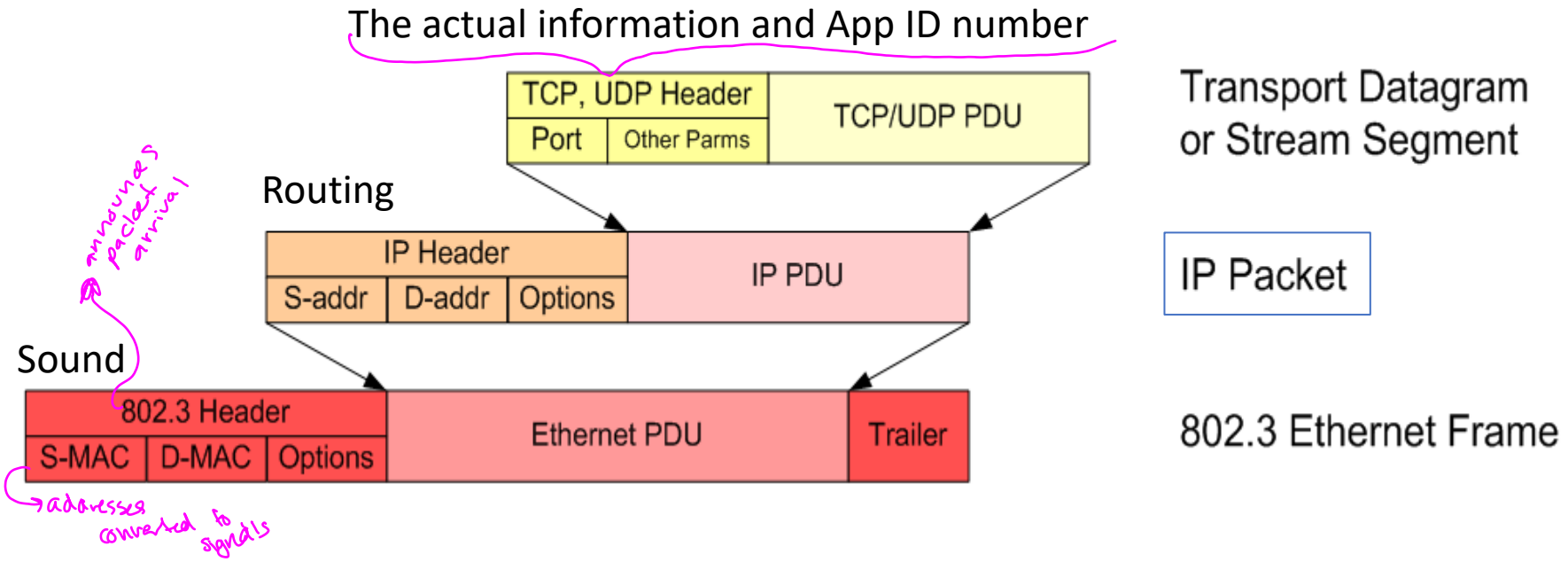
Message, or "payload", max 64K. (hence sequence number)

Actual Packet




A
messenger
"hi"
B
messenger
end-2-end
B | hi
B | hi
top world
bottom world
IP_B
IP_B
B | hi

Nested Structures



Why so much control information?

- Datagram: which App is this for?
- Routing: which machine is this for?
- Ethernet: reformatted for transmission through medium

Example

Messaging

Client 1
Client 2
9
Yo, Sam! (null)
Undefined

Source
dest
length
message

packet

Yo, Sam!

Client 1

Server

Has a choice
- Do I know client 1?
- Do I know client 2?
- Should I pass it on?

Client 2

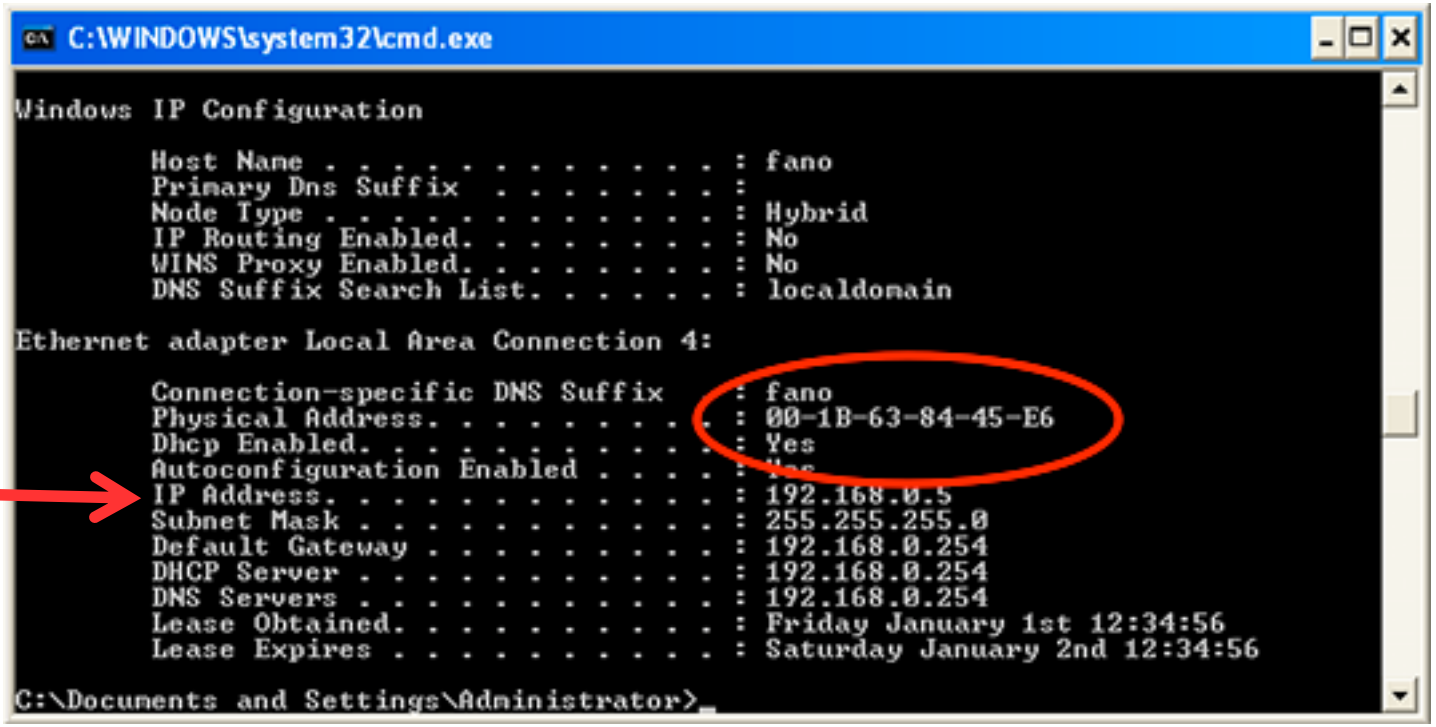
Addresses

- MAC Address
  - MAC = Media Access Control (physical)
    - ↳ physically changeable
- IP Address
  - IP = Internet Protocol (logical)
    - ↳ modifiable (software)

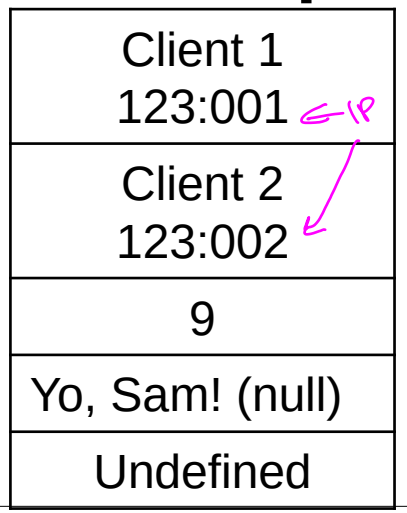
ipconfig

Every device comes with a MAC.

But you set the IP.



Example



Server

Address 123:000

Client 1

Address 123:001

Client 2 Address 123:002

Data

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Data

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

"hello" --> 104, 101, 108, 108, 111, 0

NULL, the termination character

Data to Digital

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

"hello" --> 104, 101, 108, 108, 111, 0

Binary: 104 => 1101000

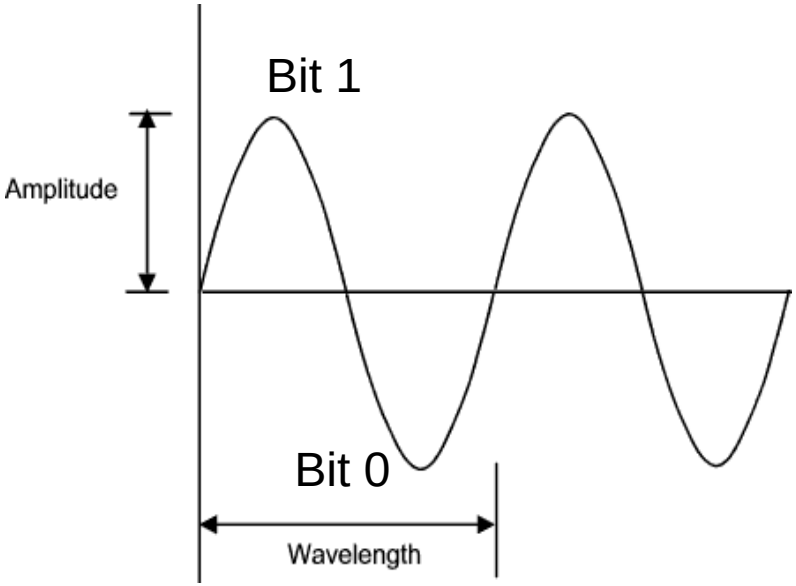
"hello" --> 1101000, 1100101, 1101100, 11001100, 1101111, 00000000

ascii to binary

Data to Signals

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]



"hello" --> 104, 101, 108, 108, 111, 0

Binary: 104 => 1101000

"hello" --> 1101000, 1100101, 1101100, 1101100, 1101111, 00000000

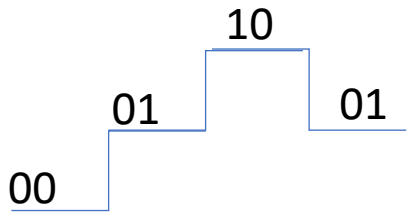
What would this look like?

binary to radio

Data Modulation

(modem)

- Instead of a simple sine wave to digitize data...
- Use multiple amplitudes: (sounds)
  - Two amplitudes: high=1, low=0
  - Four amplitudes:
    - Very high 11
    - High 10
    - Low 01
    - Very low 00
  - Eight amplitudes, 16, etc.
- Need very sensitive equipment
- Eventually not cost effective, an upper limit exists



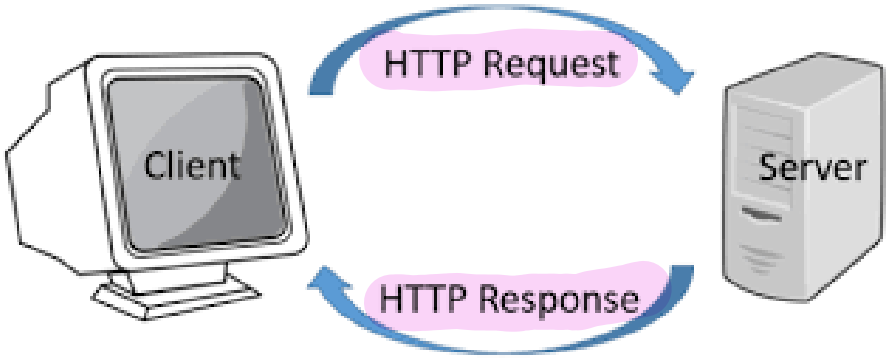
(notice data travels twice as fast)

Network Protocols

Communication Protocol

- Client A user
- Server The service you want

- Request → Asking service for something
- Response → Receiving something

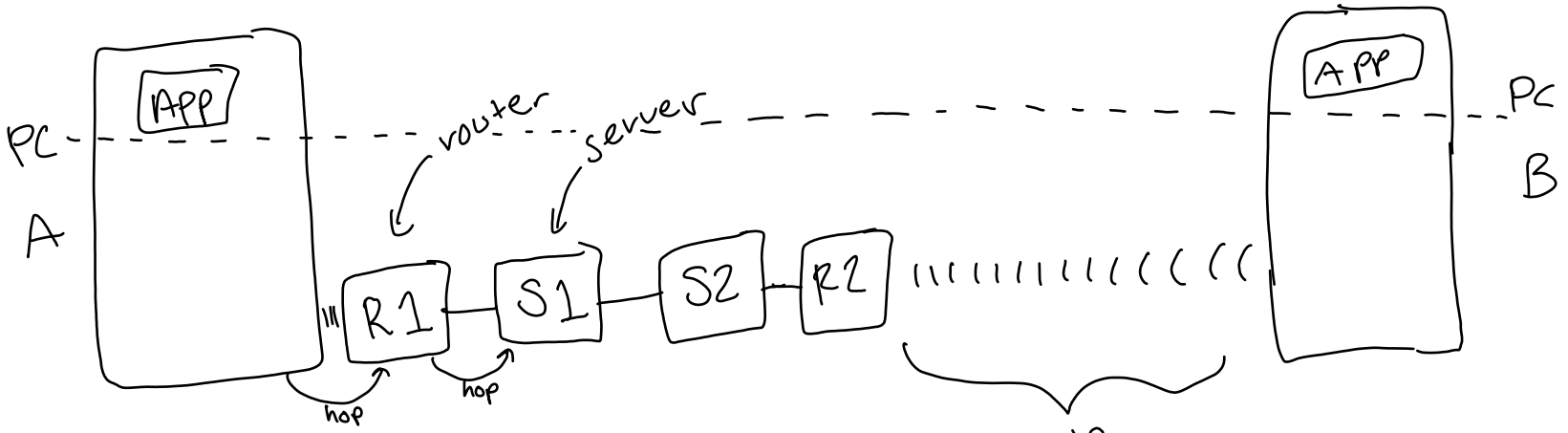


Standard Protocols

- An algorithm that describes how to transmit data.

- There are many protocols. In this course we will only look at the most common.
  1. End-to-End protocol
  2. Hop-to-Hop protocol
  3. HTTP Protocol

top = end-2-end



- sending a picture from A to B

- want as few hops as possible (5 from A→B)
- picture in 3 packets
  + header packet ← "expect 3 packets, jpg"
    ↳ like a warning for B
- B sends a confirmation back after receiving each packet from A
  ↳ Ack packet (acknowledgement)
  (ack or error)

} end-2-end

header checks for compatibility

- 3-time rule try 3 times after error
- timer before auto-generated error
  Silence is like an error

- at every hop - timer + 3 try
  ↳ send ack or err back
  ↳ no header
  ↳ one packet at a time

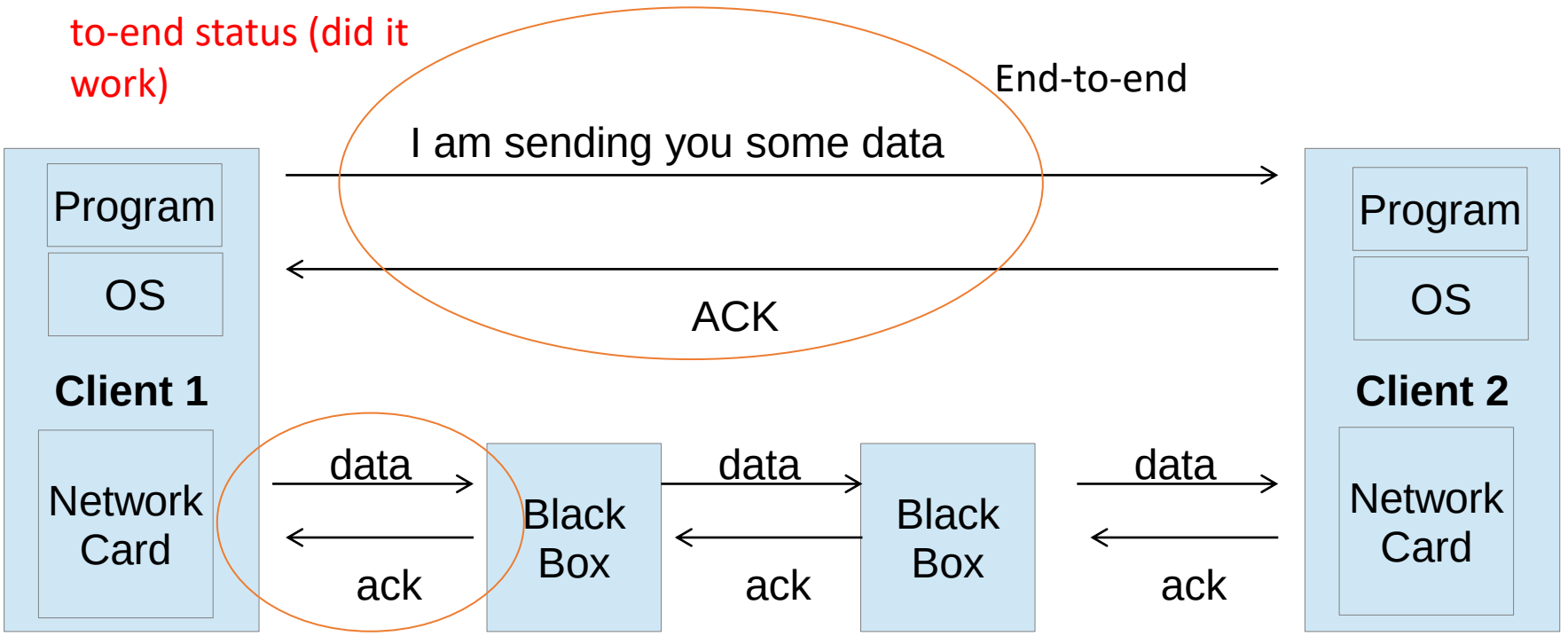
COMP 307
Principles of Web Development

End-to-End Protocol vs. Hop-to-Hop Protocol

Program waits for a success or fail code.

Two forms of control because want unsupervised communication & Want to know it worked.

OS tracks the end-to-end status (did it work)



Hop-to-hop

Network card tracks the hop-to-hop status (find the path)

Got it! Backup. Pass on.

Got it! Backup. Pass on.

Contents

End-to-End Protocol

- A protocol that informs the source and destination computer of the existence of a message.
- Source computer sends:
  - Notification that a message will be sent with:
    - Size of message integer: one packet or N packets
    - Technology destination must be able to support to read and process the message
- Destination computer sends:
  - ACK – can handle it, or ERR – cannot handle it
  - Or silence! (maybe machine is down or no path)
- Message transmission:
  - Each packet identified by a “segment” number, which is simply a sequence number from 0 to N-1.
  - Source & destination track the number of segments that arrived.

End-to-End Algorithm

SOURCE

- Try 3 times:
  - Send(# of segments)
  - Wait for ACK or fail
- Try 3 times:
  - Send a segment
  - Wait for ACK
  - Timeout? Resend
- Terminate when all segments sent & ACK received

DESTINATION

- Wait infinite
- On receive initial:
  - Check # segments
  - Send ACK or ERR
  - Start wait timer
- Wait for segment:
  - Store & sort & ACK
  - Timeout? Prompt
    - 3rd time prompt or Fail.
  - Corrupt? Err

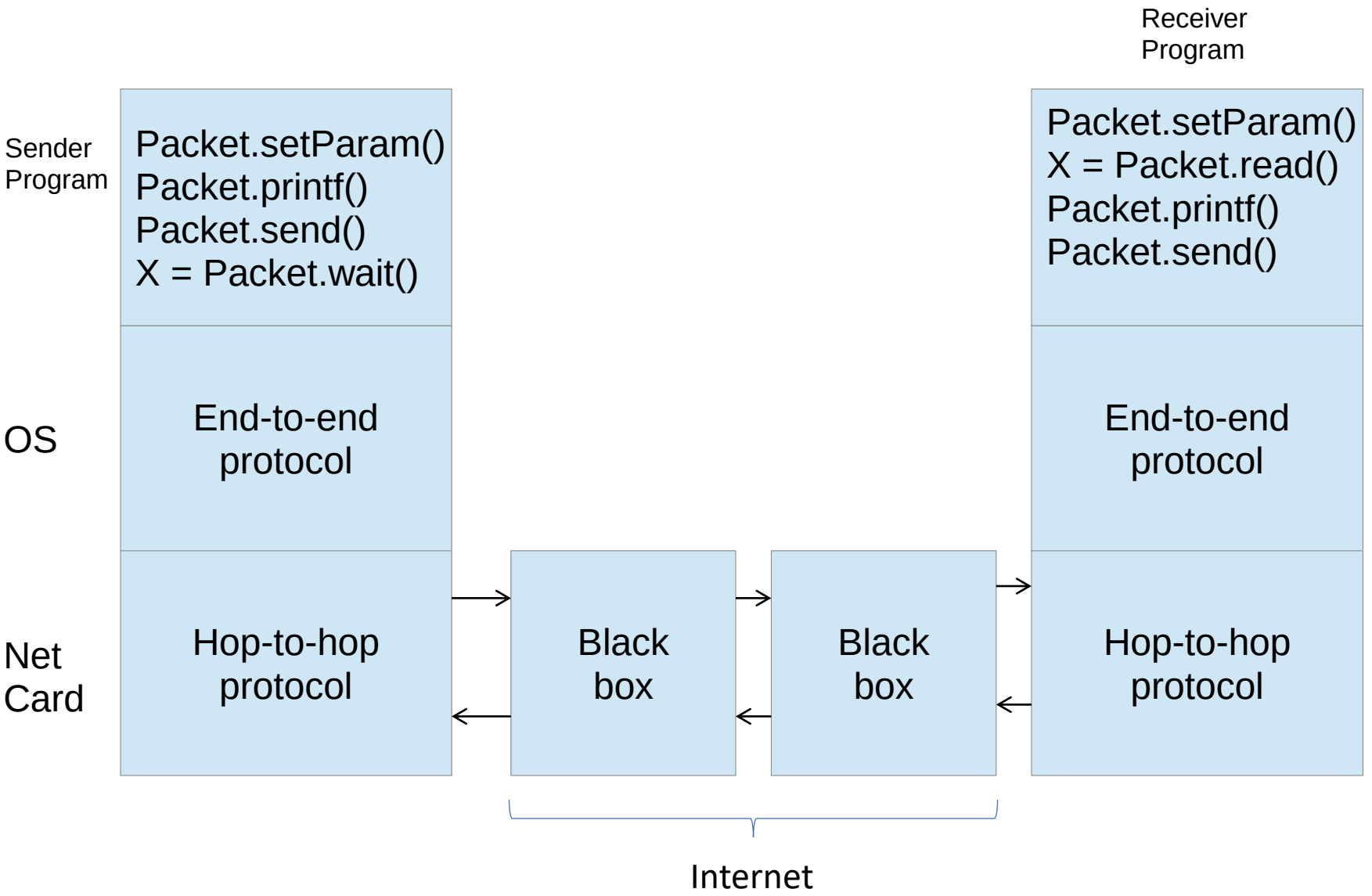
Hop-to-Hop Protocol

- The network cards and black boxes are connected without the knowledge of the OS. This is called the Internet backbone
- A packet (segment) must pass through these intermediate computers from source to destination. The OS is not involved.
  - This game of “hot potato” needs to be managed?
  - Traffic? (is there a better path?)
  - Lost? (the packet never arrived!)
  - Damaged? (unable to understand the packet)

Hop-to-Hop Algorithm

- Actors:
  - Sending Hop, Receiving Hop, Packet
  - Packet has source & destination address
- Sending Hop:
  - Looks at packet's destination address & mode
  - If mode == broadcast, then sends to all connected hops
  - If mode == route, then check network map for destination and send to the most available hop
  - Make a backup of packet, and wait for ACK or ERR
  - If ERR or TIMEOUT, then resend (repeat 3 times, fail)
- Receiving Hop:
  - Looks to see if it can read packet, no: ERR
  - Convert itself to a sending hop

Software point-of-view



Example

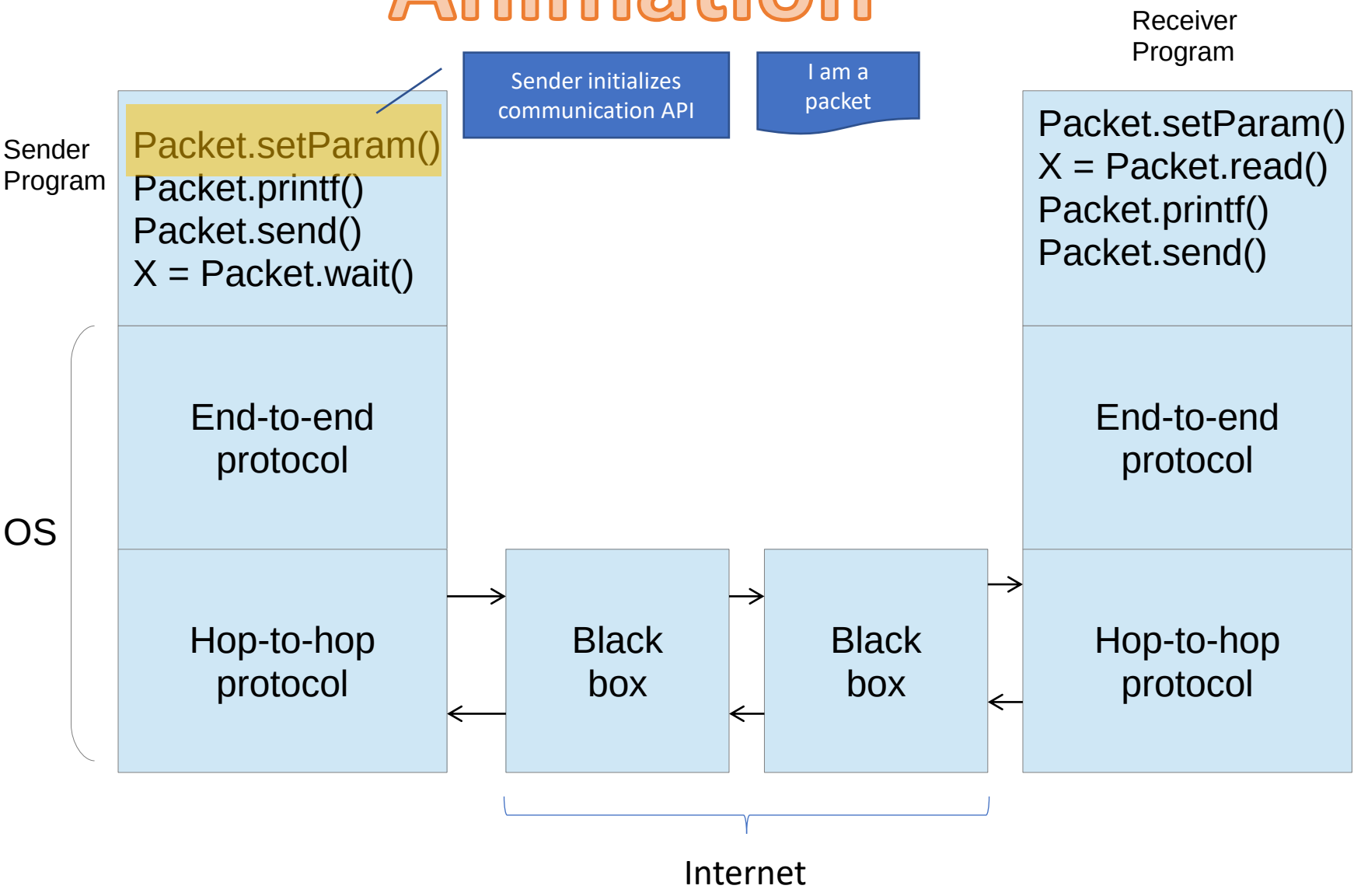
- Client wants to send a single message to server. Message is 1 packet long.
- Client – Hop1 – Hop2 – Server
- What end-to-end and hop-to-hop packets are sent to move the message from client to server?

time

Client	Hop 1	Hop 2	Server

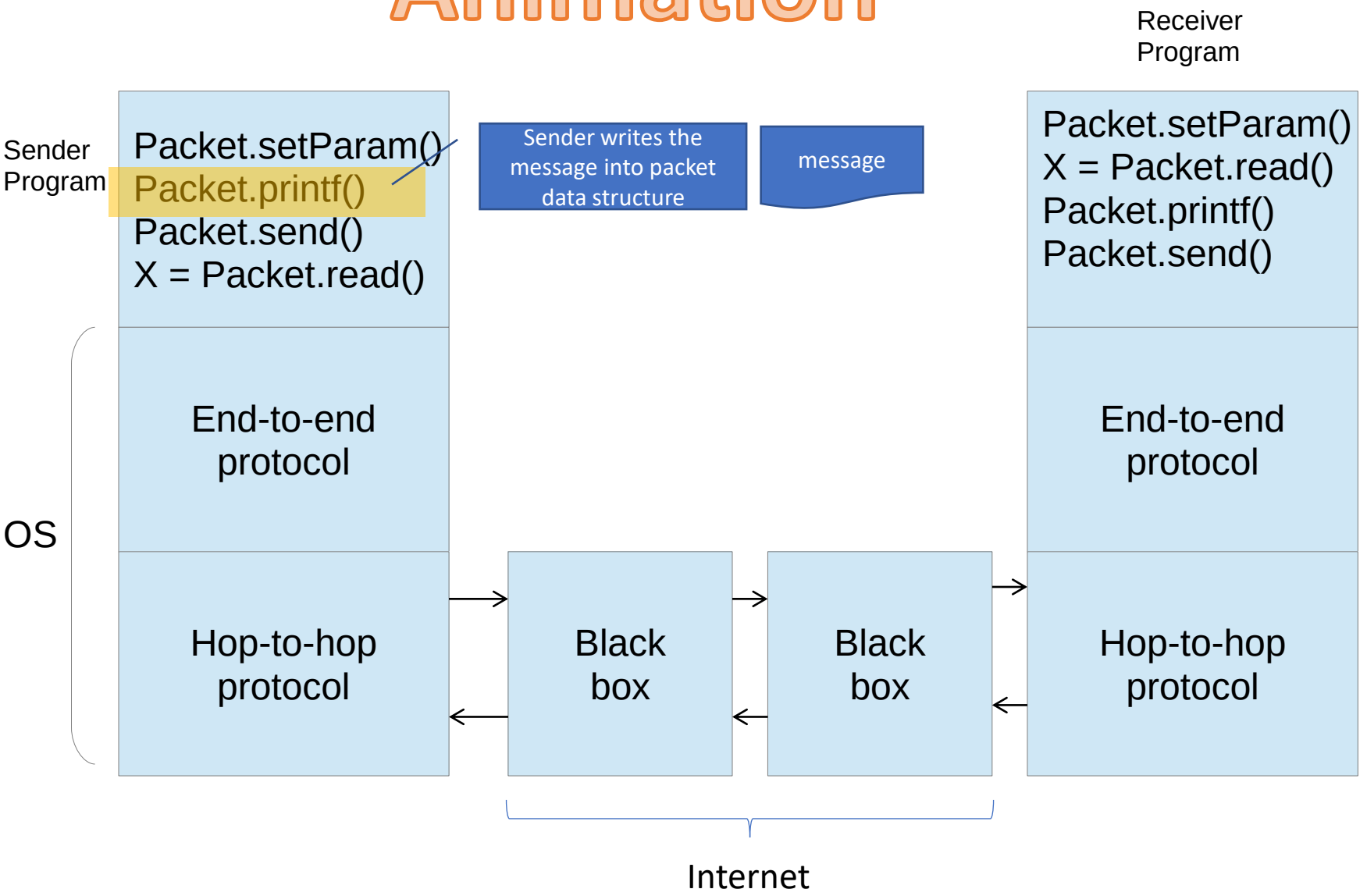
Software point-of-view

Animation



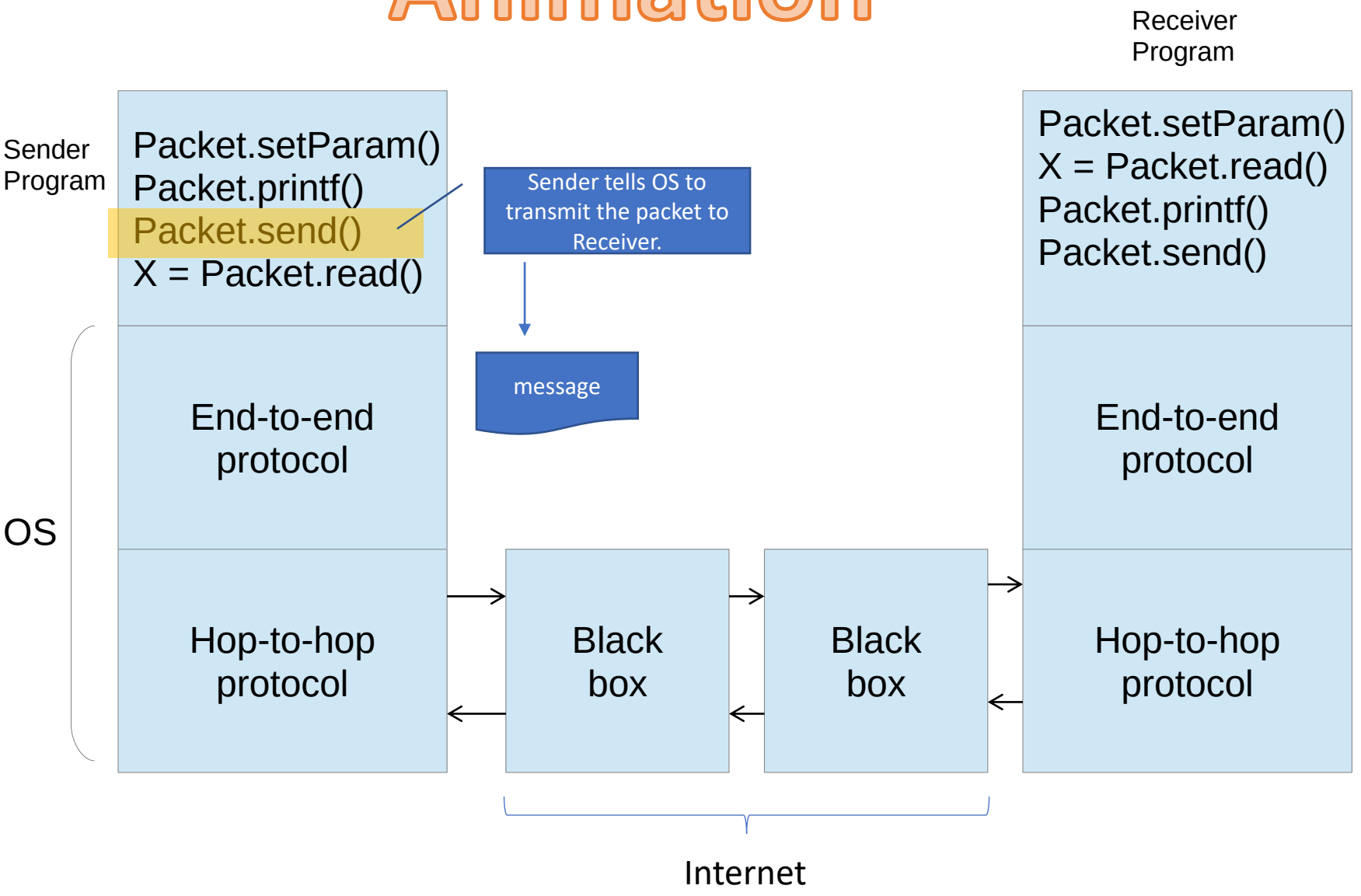
Software point-of-view

Animation



Software point-of-view

Animation



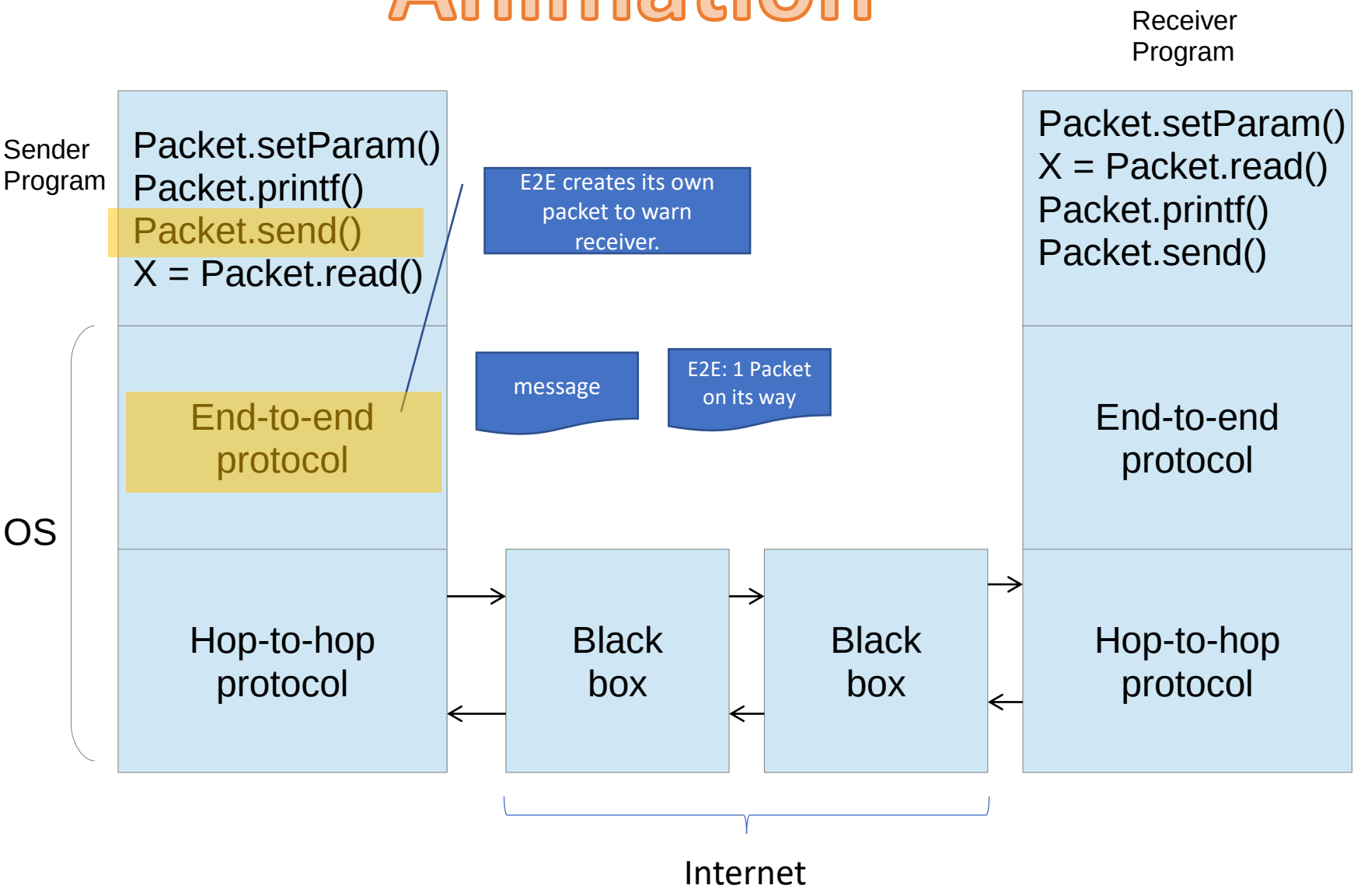
COMP 307
Principles of Web Development

Contents

Unsupervised Networks
HTTP Protocol

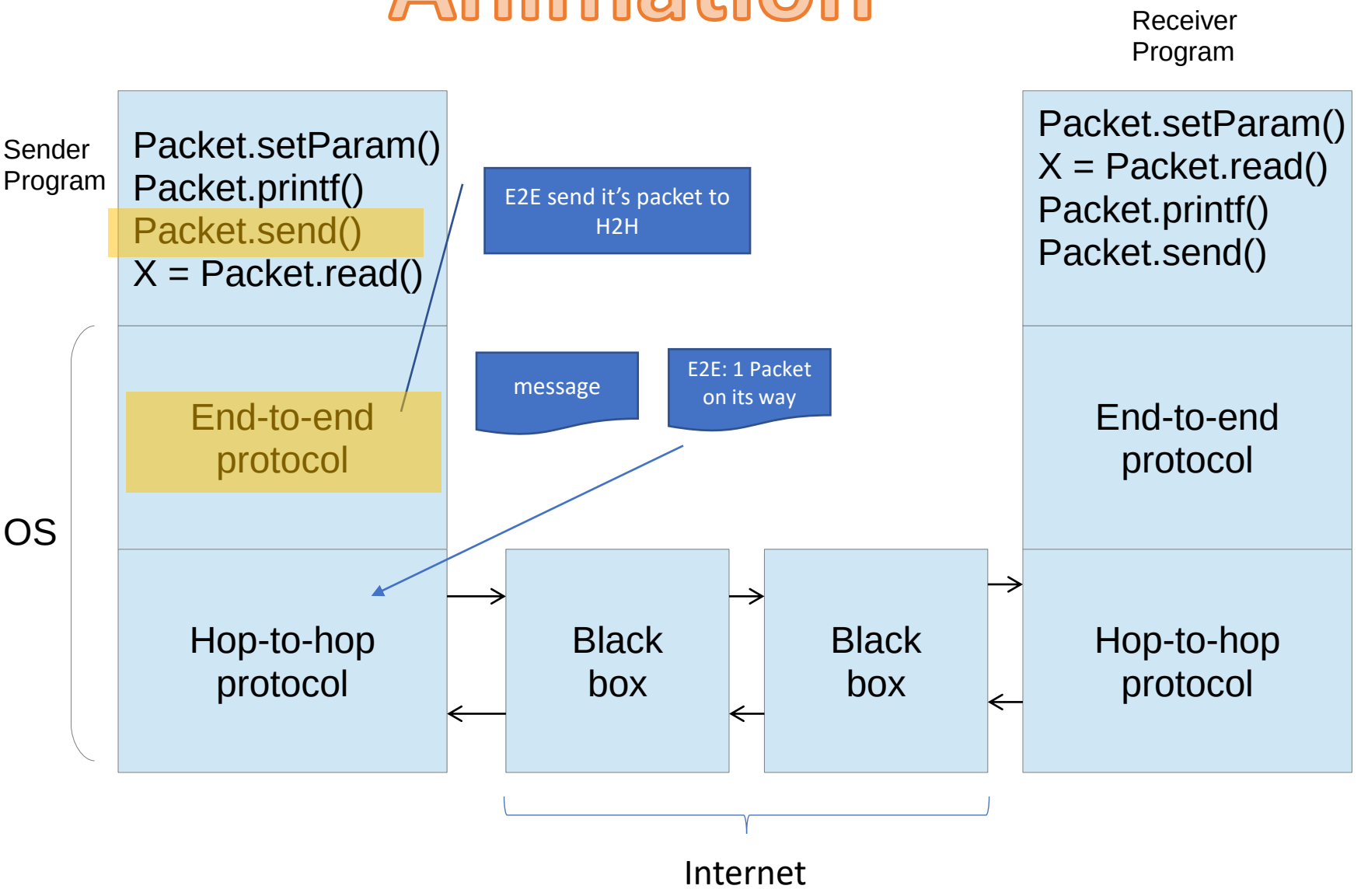
Software point-of-view

Animation



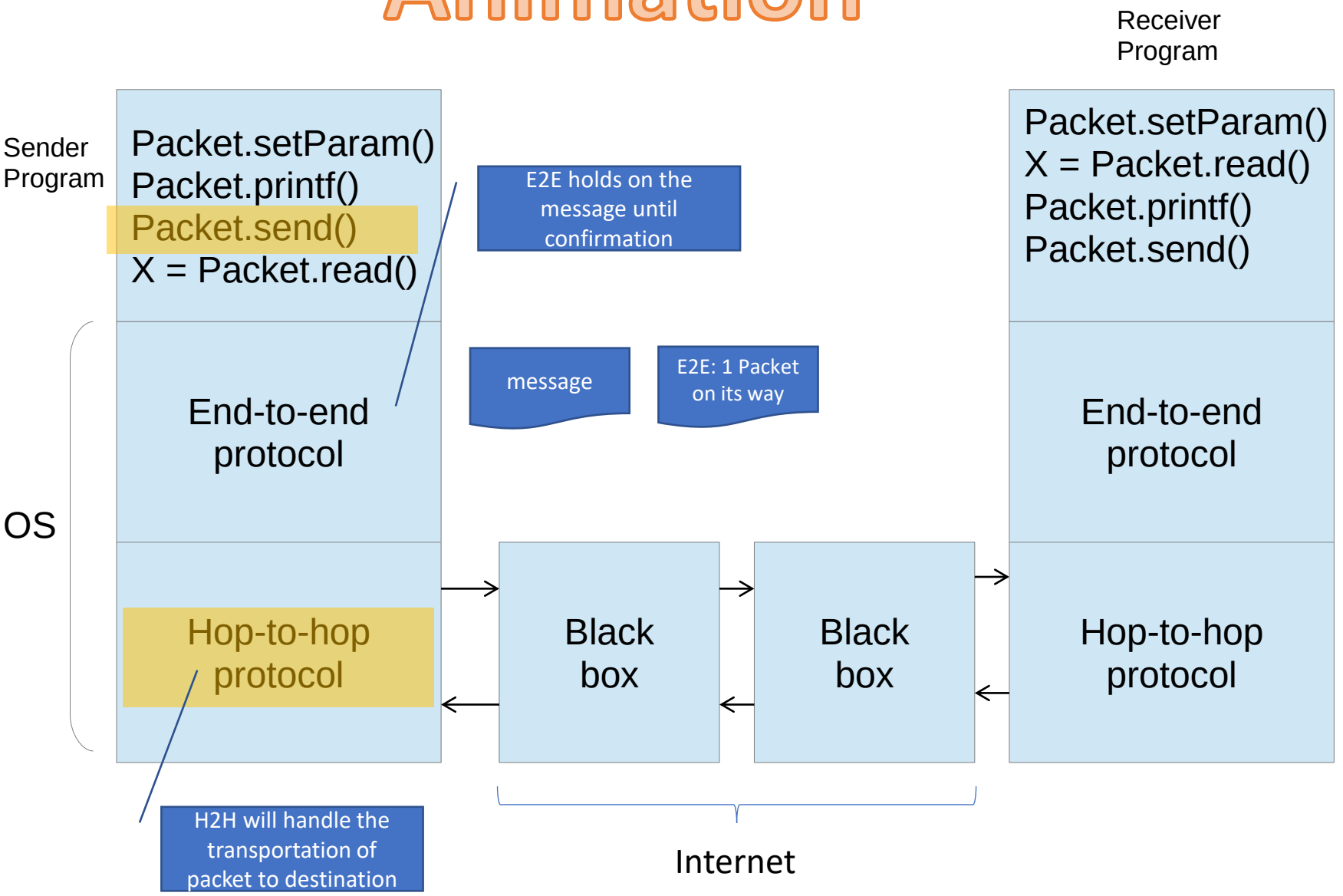
Software point-of-view

Animation



Software point-of-view

Animation

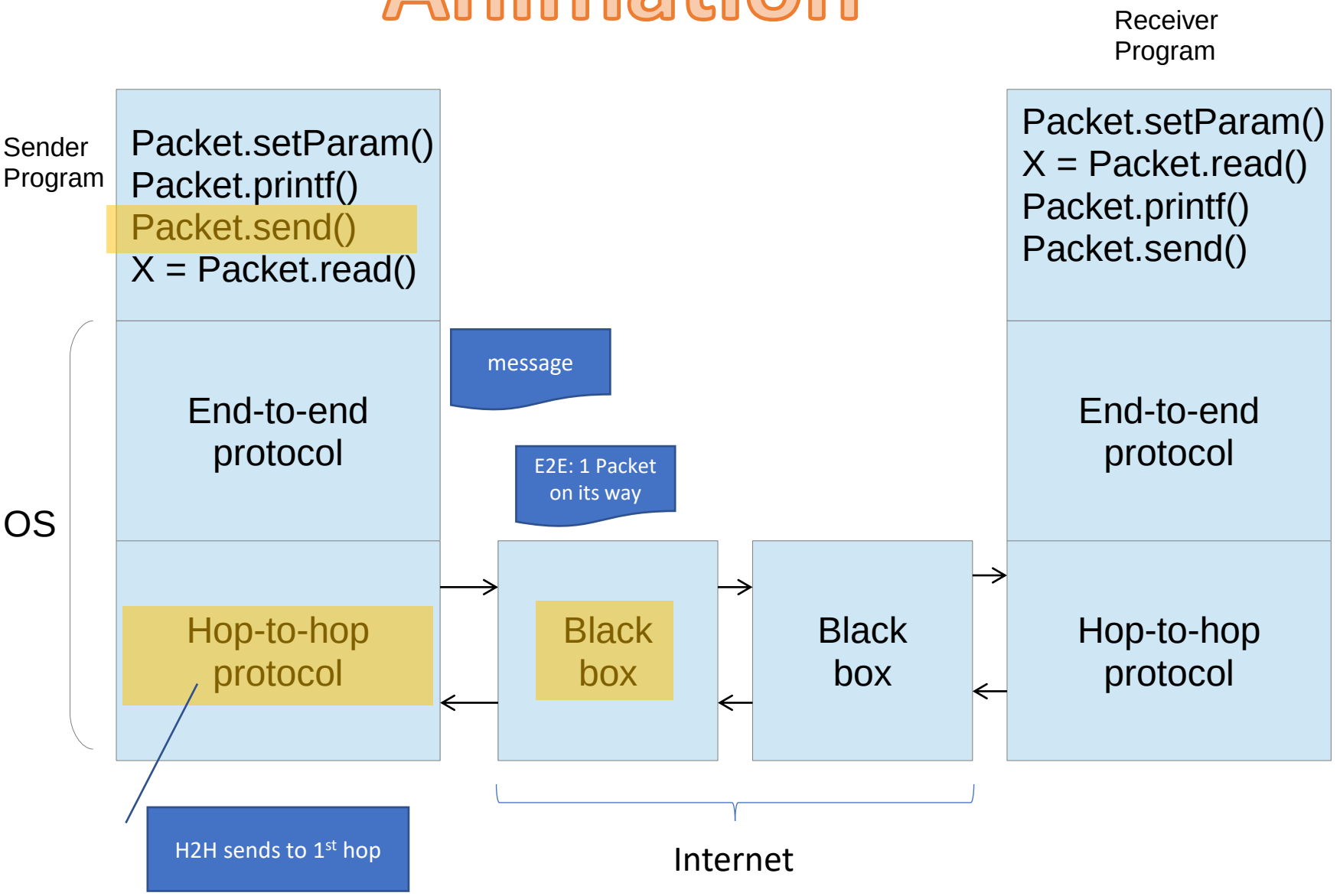


COMP 307
Principles of Web Development

Contents

Software point-of-view

Animation

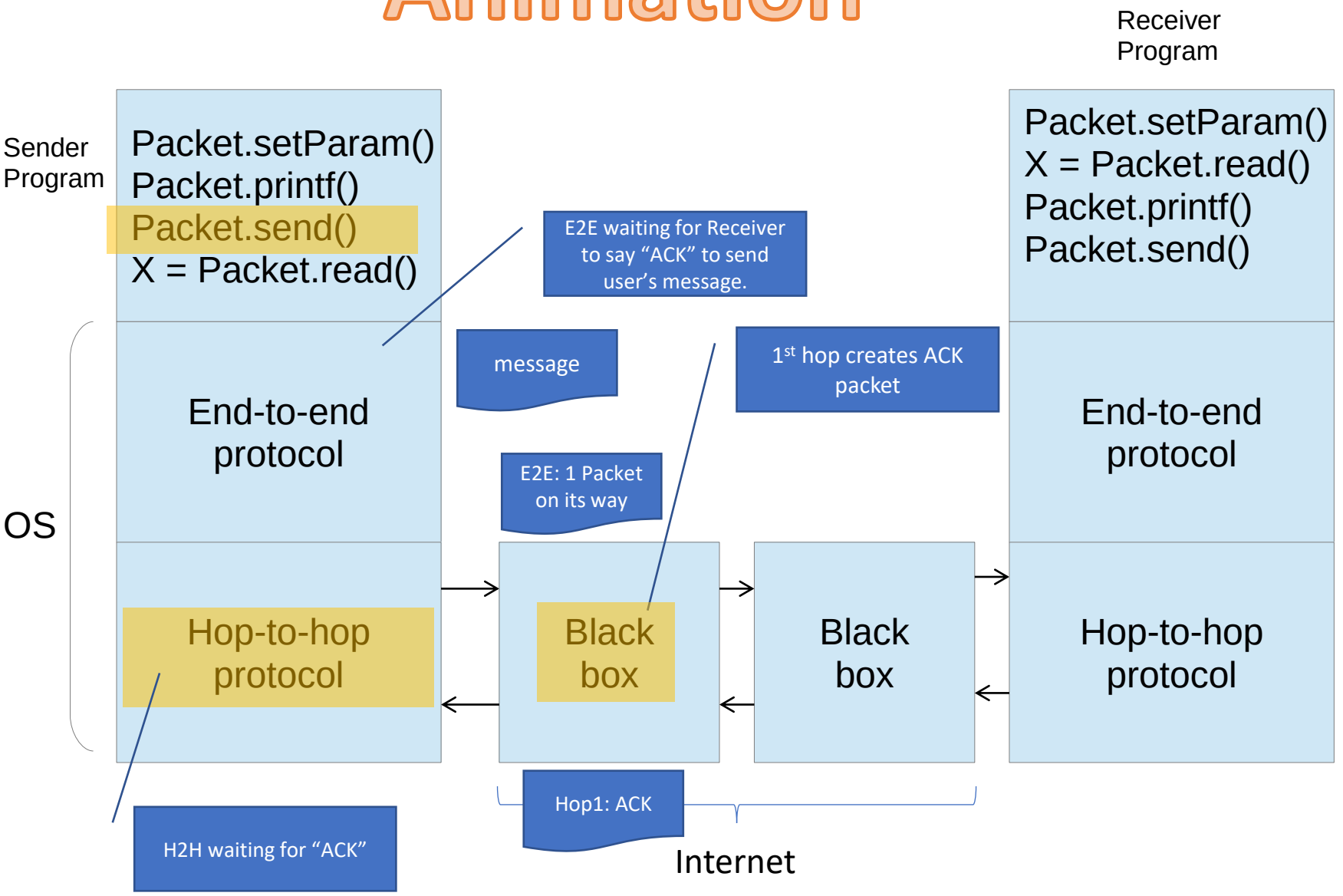


COMP 307
Principles of Web Development

Contents

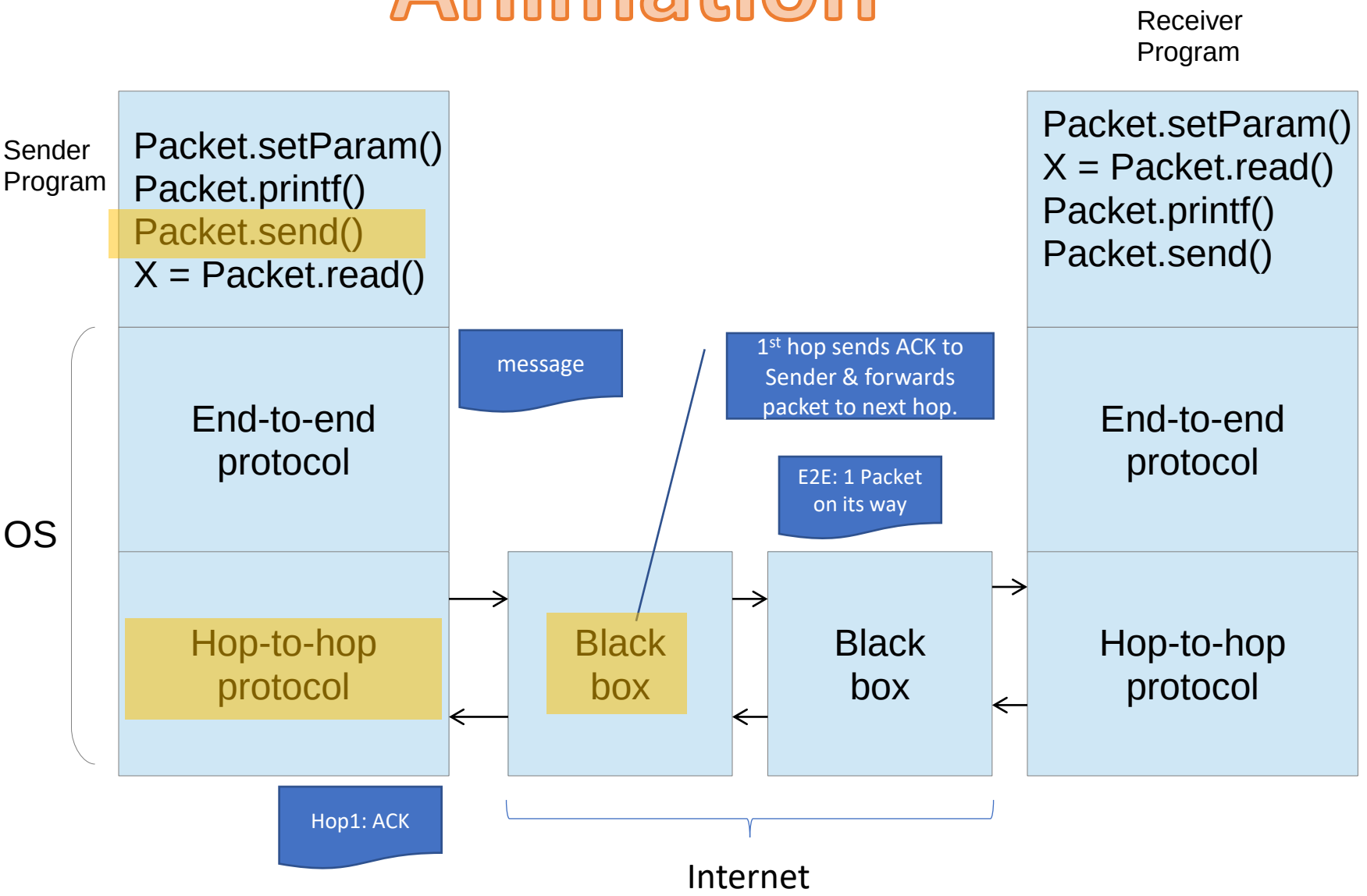
Software point-of-view

Animation



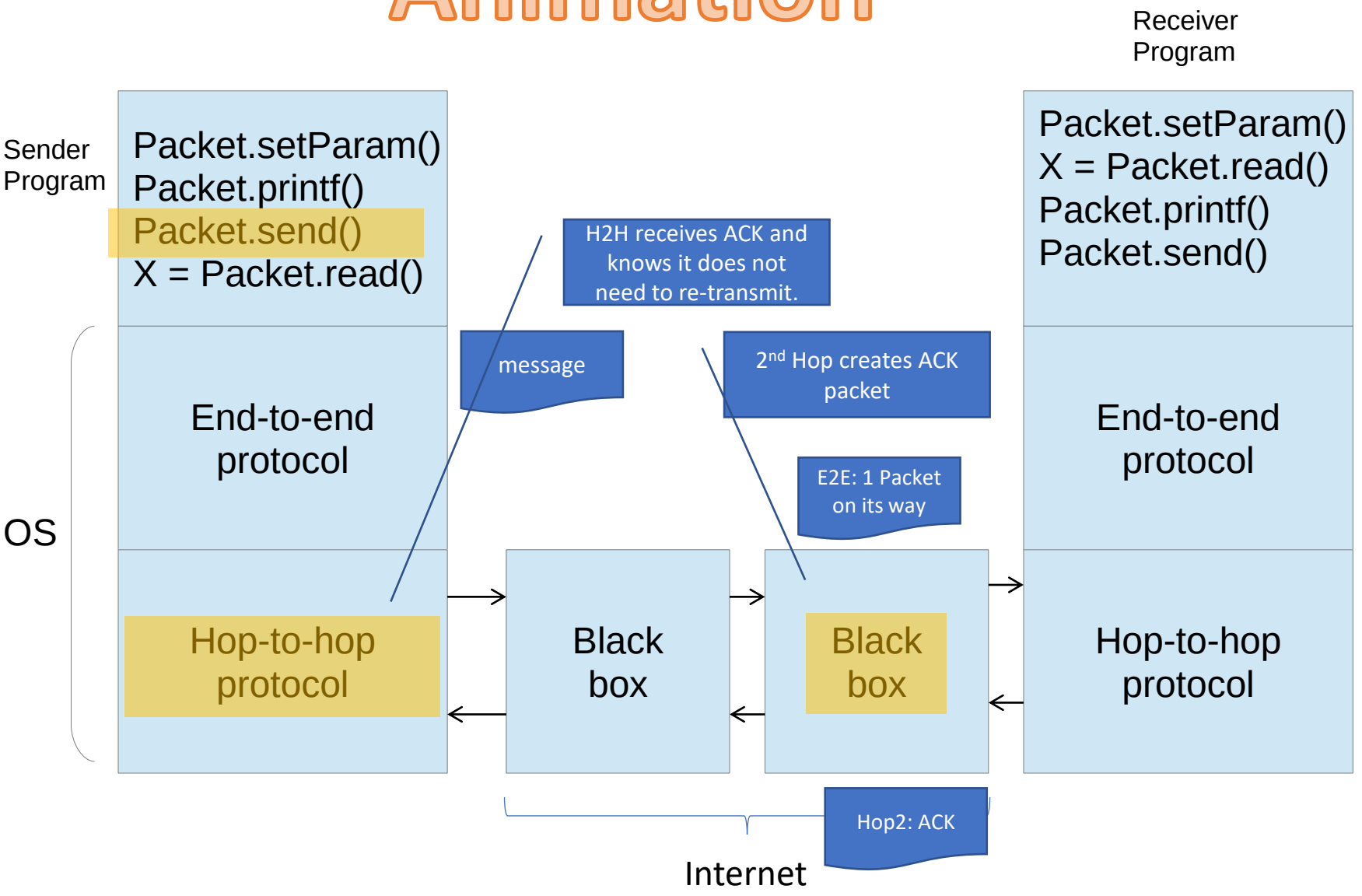
Software point-of-view

Animation



Software point-of-view

Animation

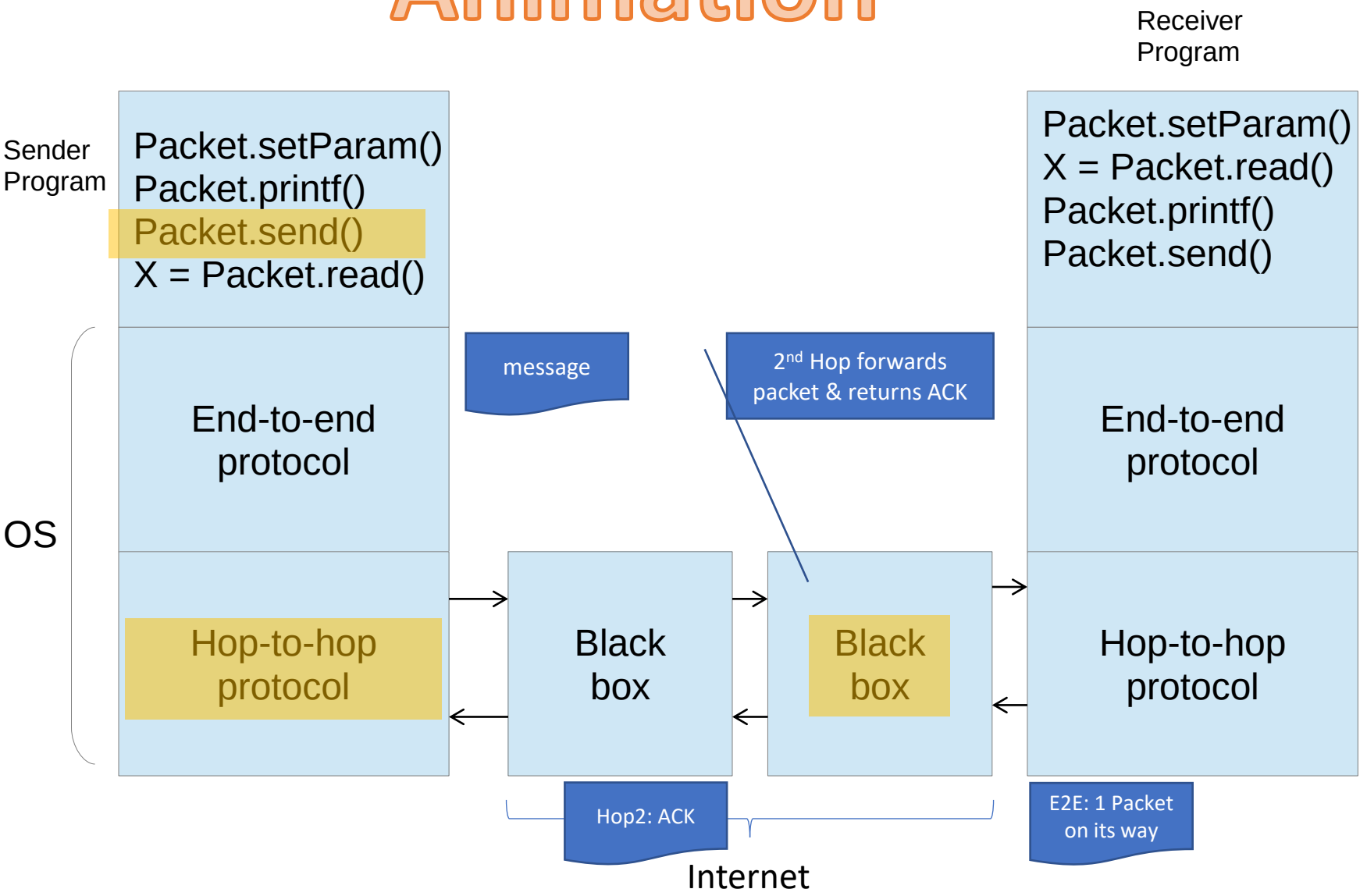


COMP 307
Principles of Web Development

Contents

Software point-of-view

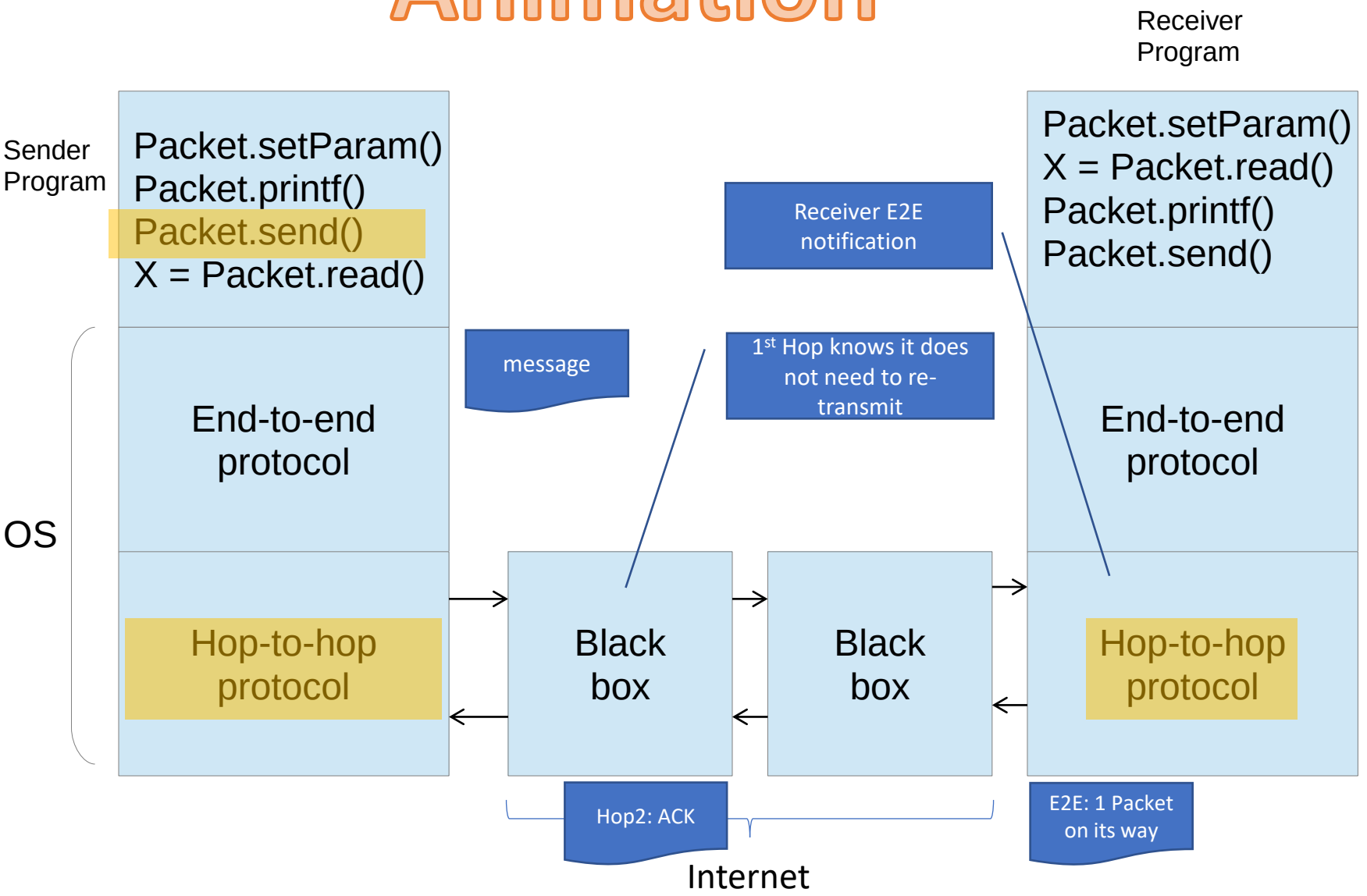
Animation



COMP 307
Principles of Web Development

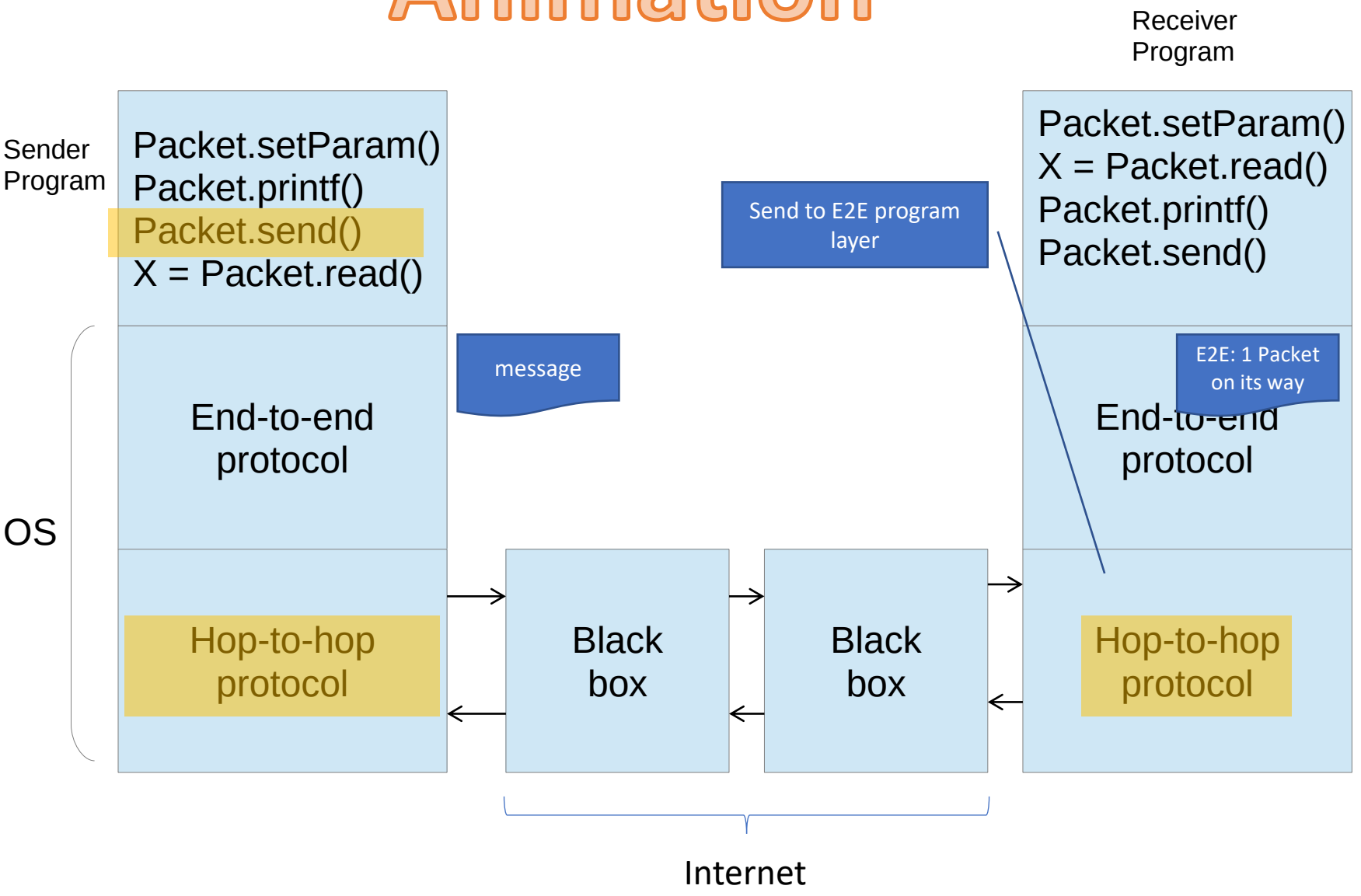
Software point-of-view

Animation



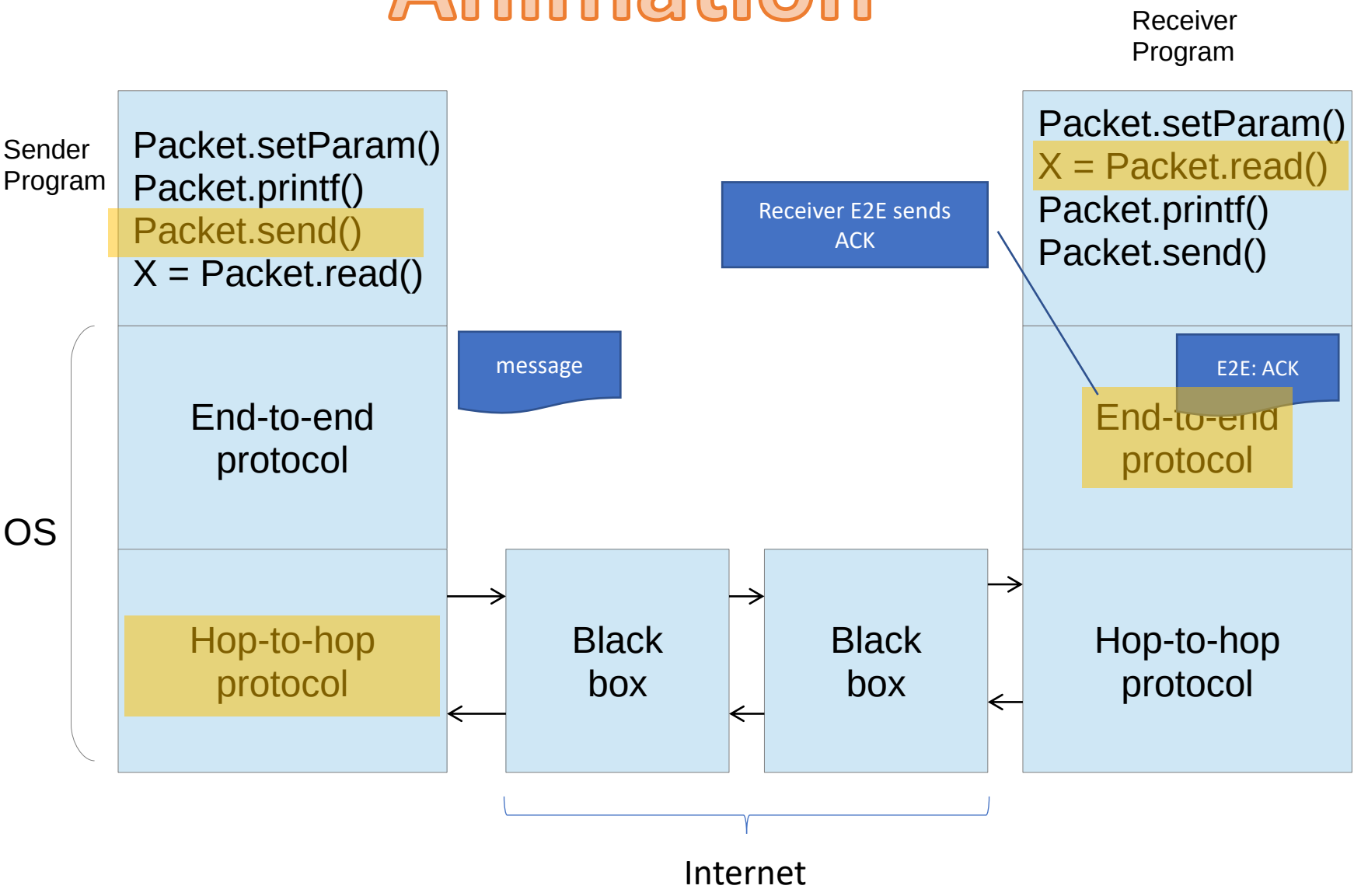
Software point-of-view

Animation



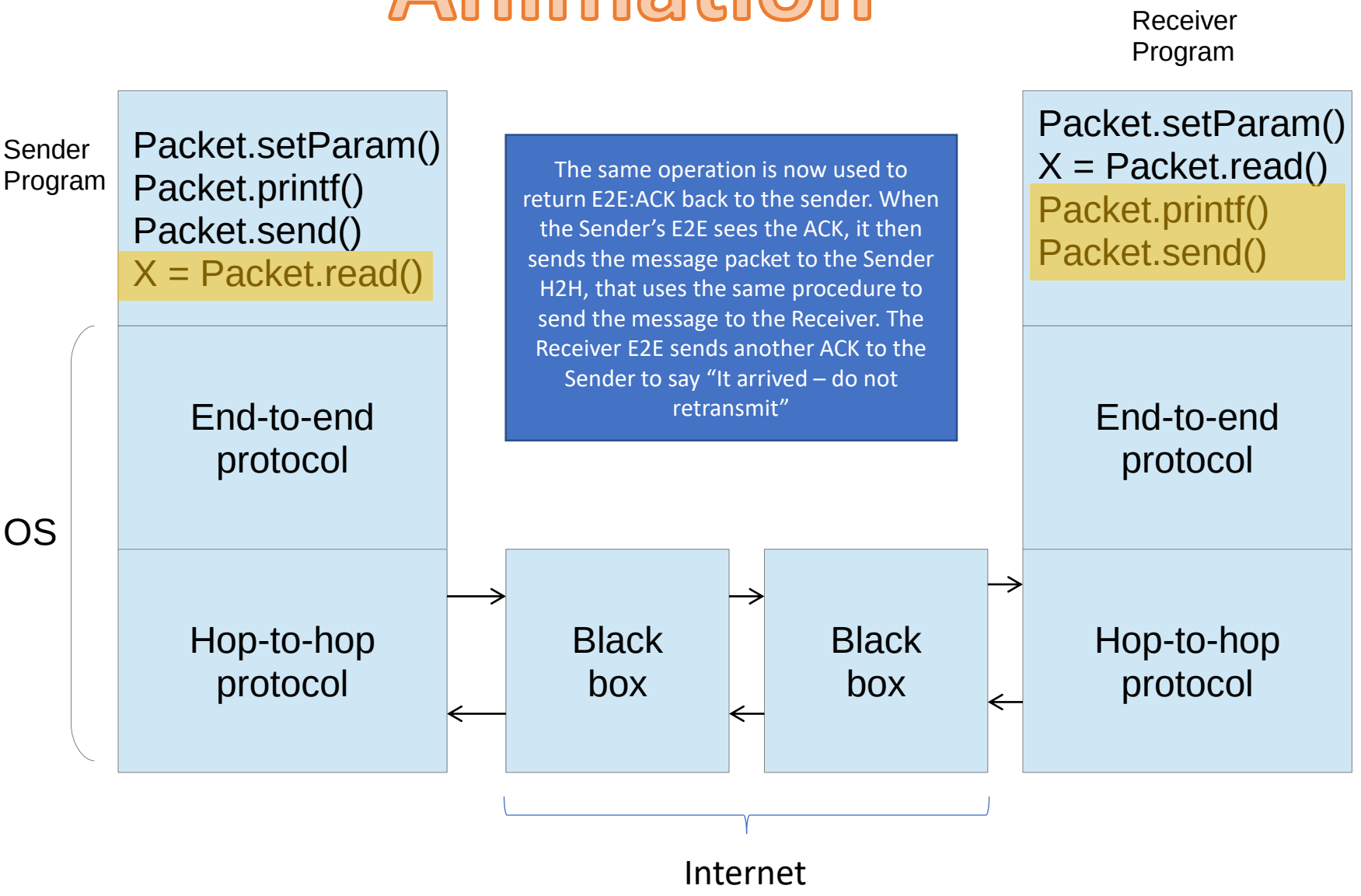
Software point-of-view

Animation



Software point-of-view

Animation



COMP 307
Principles of Web Development

how the internet sends messages to each other

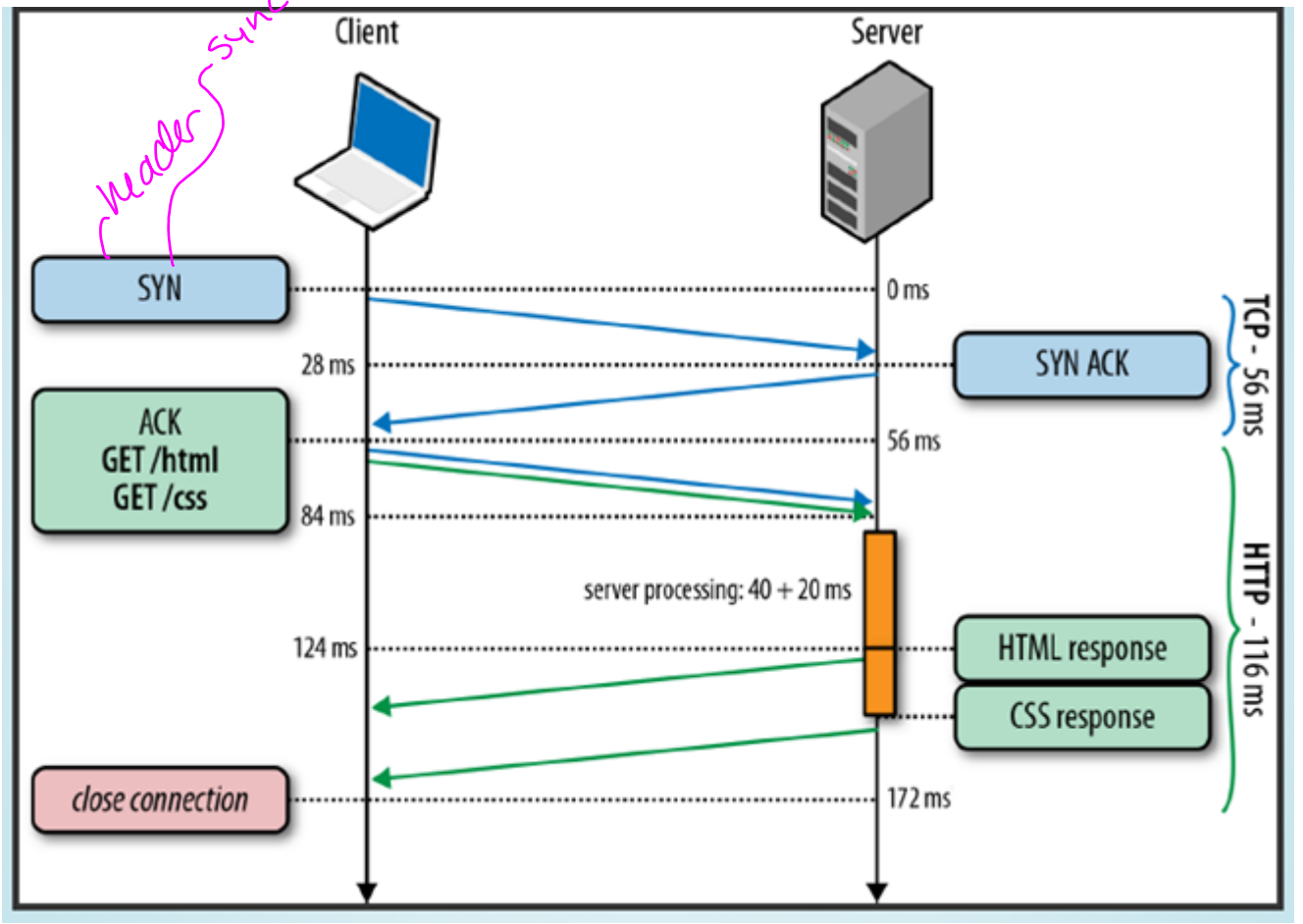
TCP/HTTP Protocol

Transmission Control Protocol / Hyper Text Transfer Protocol

Communication Diagram

synchronization

header



Connect & query

Reply

End

Query and then close connection (forget) – "stateless"

Need to reconnected for each query!!

hello, no goodbye

no acknowledgment at end

HTTP Protocol

Request method	RFC	Request has payload body	Response has payload body	Safe	Idempotent	Cacheable
GET	RFC 7231	Optional	Yes	Yes	Yes	Yes
HEAD	RFC 7231	Optional	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	Optional	Yes	No	Yes	No
CONNECT	RFC 7231	Optional	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	No

GET	Get data from server	DATA
POST	Save data to server (new data)	
PUT	Modify existing data on server (overwrite)	
PATCH	Modify existing data on server (edit)	
DELETE	Delete data on server	
HEAD	Get network information	SYSTEM
CONNECT	Establish a connection with the server	
OPTIONS	Ask for server features	
TRACE	Debug the communication pathway to server	

Example

Assuming a browser uses HTTP requests and someone types: www.youtube.com

What HTTP requests and ACKs pass between the client and the Service Provider? In what order?

- Use a communication diagram

Wireshark Demo

- Http://wireshark.org
  - Download and install Wireshark
  - A great way to see real packets
  - A great way to see other people's packets!

Prepare for next class

- Assignment
  - Start Mini 1
- On your own
  - Try to redo the in-class demo of Wireshark
  - Understand the Hop-to-Hop and End-to-End protocol
- Lab
  - Lab A (Wireshark) – check TA schedule