NETWORKING REVIEW (PROTOCOLS)

UT CS361S – Network Security and Privacy

Spring 2021

Lecture Notes

COMPUTING 1960-1980 (ISH)

NETWORK

This Photo by Unknown Author is licensed under CC BY-SA-NC

"DUMB" TERMINAL

MAINFRAME

COMPUTING 1980-2000 (ISH)

This Photo by Unknown Author is licensed under CC BY-NC

COMPUTING 2000 – PRESENT

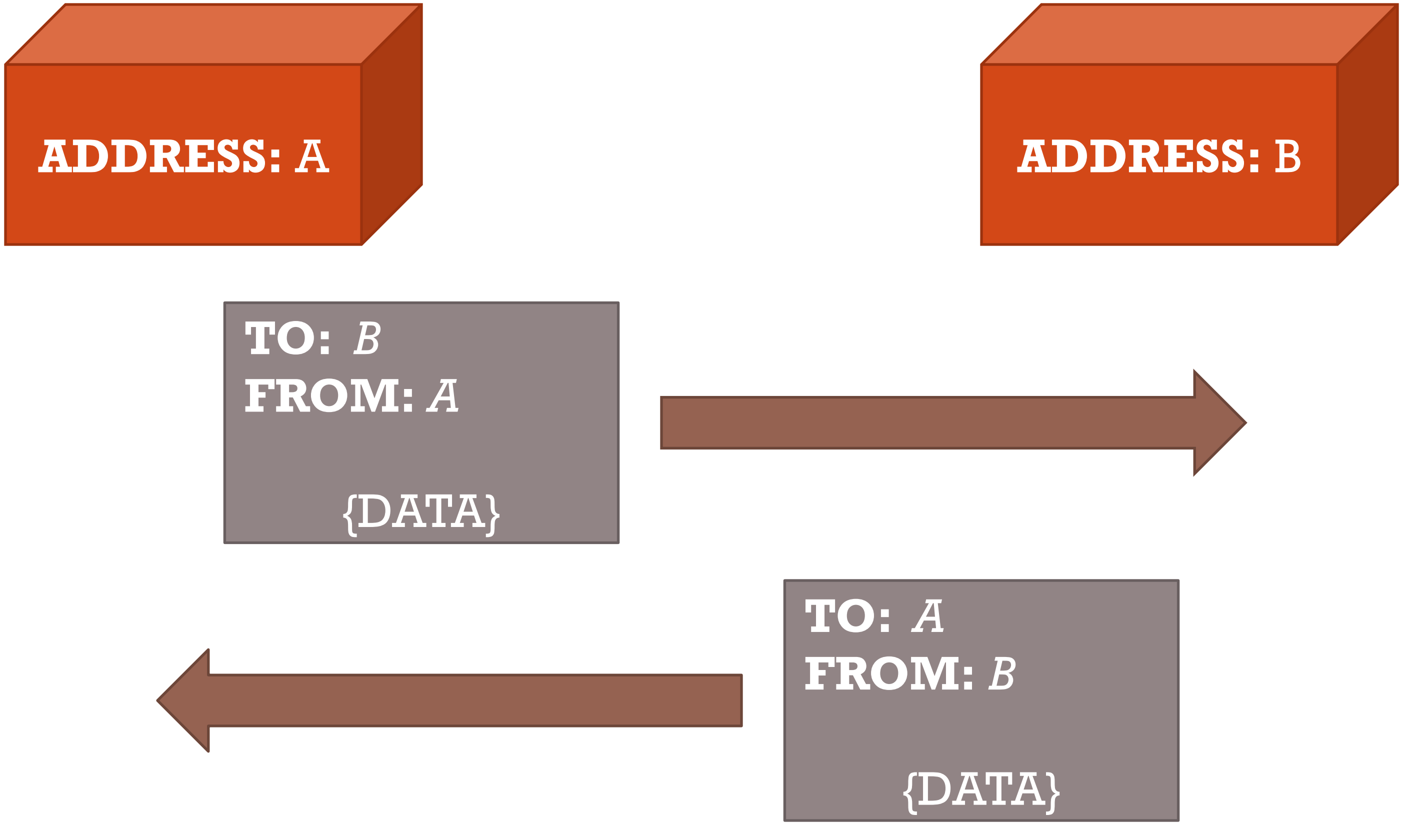
This Photo by Unknown Author is licensed under CC BY

NETWORK

This Photo by Unknown Author is licensed under CC BY-NC-ND

SIMPLIFIED NETWORK COMMUNICATION

- Two participants; we will call each one ***a “node”***
- Data is transmitted between participants in ***packets*** (chunks)
- Packet includes metadata, usually prepended as a ***header***
- Header includes ***sender’s address*** *and* **receiver’s address**
- Receiver can “reply” to sender’s packets by reversing addresses



ADDRESS: A

ADDRESS: B

TO: *B*
FROM: *A*

{DATA}

TO: *A*
FROM: *B*

{DATA}

ALSO, NETWORK DISCOVERY

- Many communication patterns require ***discovery***
- Either, the discovery of the existence of nodes
- And/or discovery of how to communicate with a specific node

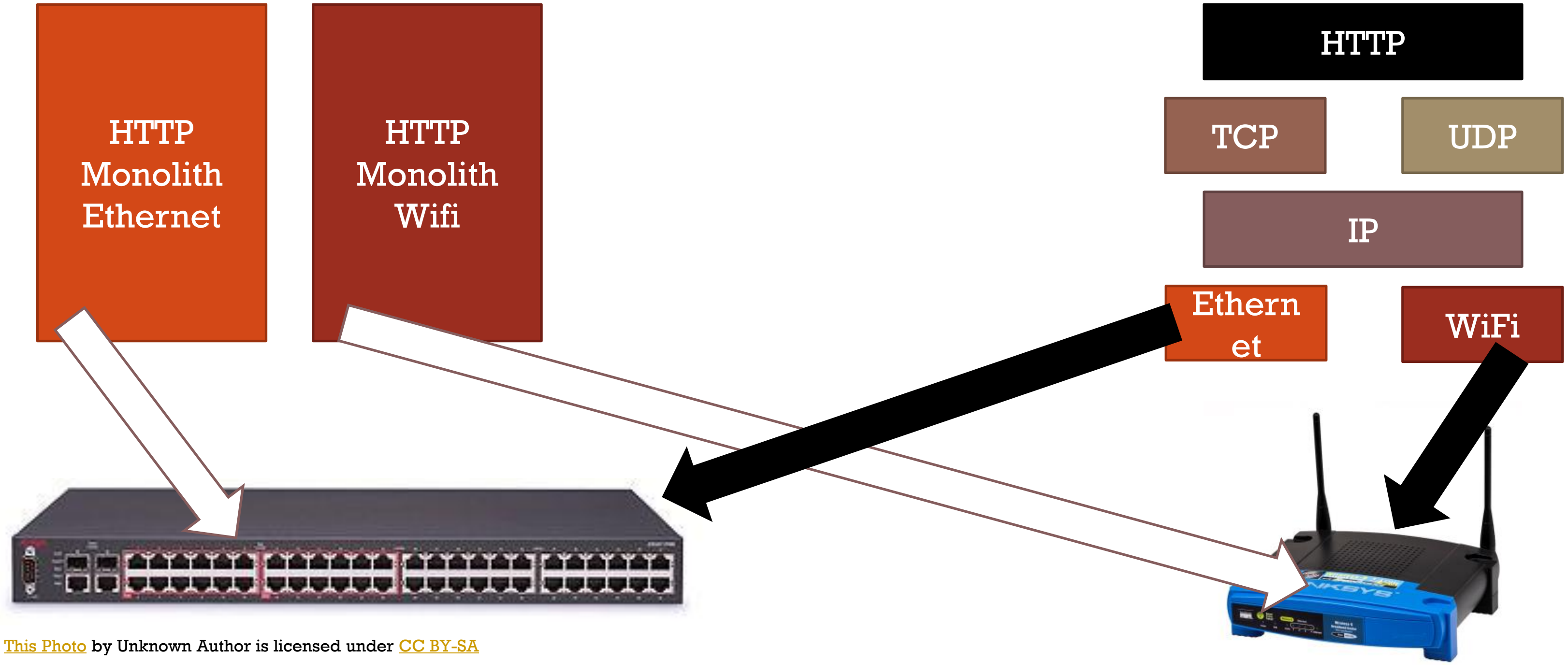
WHAT IS A PROTOCOL?

- The set of rules that govern the interaction of two or more parties
- In networking, it defines how two nodes communicate
  - When
  - What *(including message structure)*
  - How
- ***Certain outcomes are guaranteed when the rules are followed***

PROTOCOL *STACKS*

- Object-oriented design!
  - Modularity
  - Abstraction
  - Information hiding
- Protocols are designed in an object-oriented fashion
  - Protocols are combined to solve more complex problems
  - Each protocol should focus on one purpose/goal (High Cohesion)
  - Different component protocols can be swapped (Low coupling)
- We call a group of protocols that work together a *protocol stack*
- In networking, a *network protocol stack* or a *network stack*

MONOLITHIC VS MODULAR



HTTP Monolith Ethernet

HTTP Monolith Wifi

HTTP

TCP

UDP

IP

Ethernet

WiFi

This Photo by Unknown Author is licensed under CC BY-SA

This Photo by Unknown Author is licensed under CC BY-SA

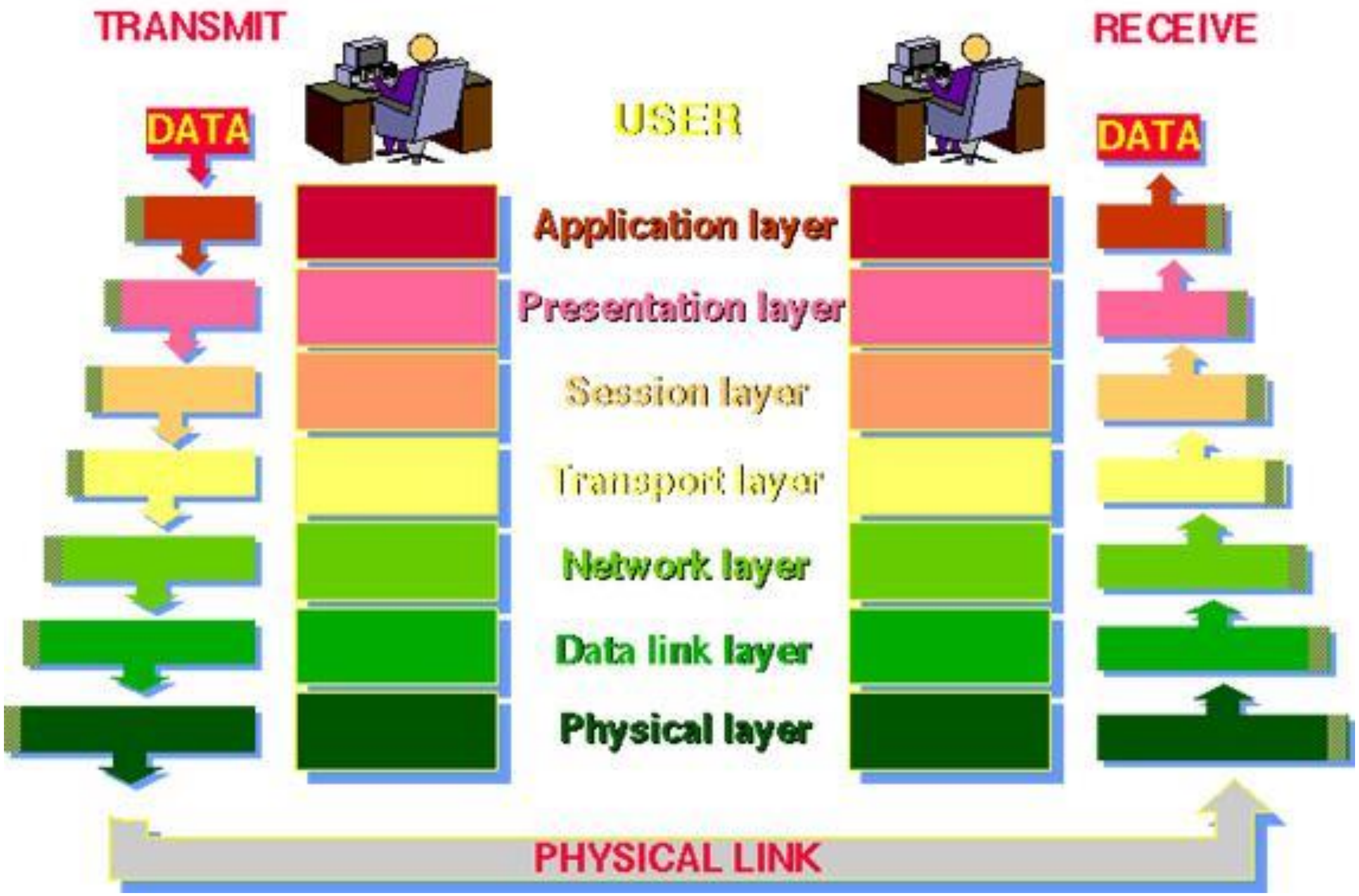
OTHER PROBLEMS WITH MONOLITHIC

- No separation of user/kernel space components
- Code cannot be reused; code bloat
- NxM combinations
- Patching nightmare
- Testing limitations
- List goes on and on

OSI MODEL

- Good object-oriented design is implementation independent
- ***Conceptual guide*** for any given network stack ***implementation***
- It has seven layers:
  - 7: Application
  - 6: Presentation
  - 5: Session
  - 4: Transport
  - 3: Network
  - 2: Data Link
  - 1: Physical

THE 7 LAYERS OF OSI



TRANSMIT

RECEIVE

DATA

USER

DATA

Application layer

Presentation layer

Session layer

Transport layer

Network layer

Data link layer

Physical layer

PHYSICAL LINK

THE OSI MODEL IN PRACTICE

- Like most OO-designs, the abstraction often breaks down
- The TCP/IP stack really only uses the following layers:
  - Application (Layer 7; example: HTTP)
  - Transport (Layer 4; TCP)
  - IP (Layer 3; IP)
  - Data Link (Layer 2; example: Ethernet)
- Some breakdown in information hiding, abstractions, etc
- NOTE: It's common to just refer to a layer by it's number (e.g., a layer-4 protocol)

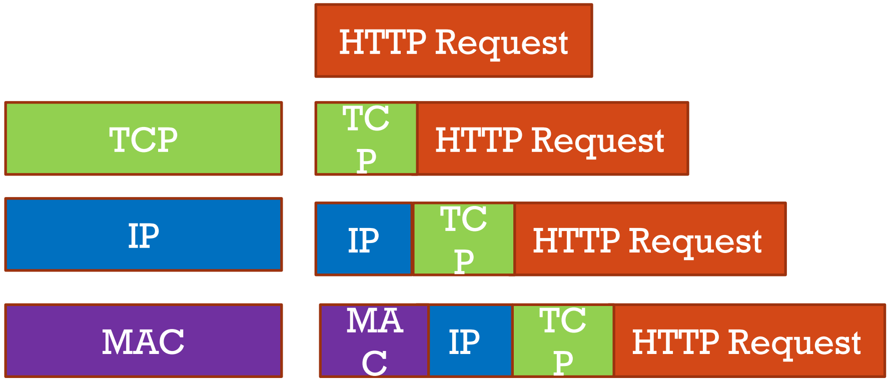
TCP/IP STACK

- For our purposes, we will focus on TCP/IP and TCP/IP-like stacks
- The TCP and IP layers are fixed for layers 3 and 4 (***hourglass***)
- But layers 7 and 2 vary widely
- Millions of networked applications work over TCP/IP at layer 7
- Many layer 2 protocols such as WiFi, Ethernet
  - Networked applications work over WiFi or Ethernet without any change
  - Sometimes called a MAC protocol (Media Access Protocol)
  - TCP/IP work over a walkie-talkie with an appropriate MAC protocol

HOW DOES DATA MOVE IN A STACK?

- To send, data is inserted (pushed) at the top-most protocol
- The receiving protocol
    - Processes the data, potentially splitting, recoding, etc
    - Derives one or more chunks of output data
    - Metadata added to each chunk (usually a header)
    - Each chunk, along with the meta-data is a "packet"
    - The packet is inserted (pushed) down to the next layer
- On the receiving side, the process is reversed, starting at bottom

TCP/IP STACK SEND EXAMPLE



DIVISION OF LABOR IN TCP/IP

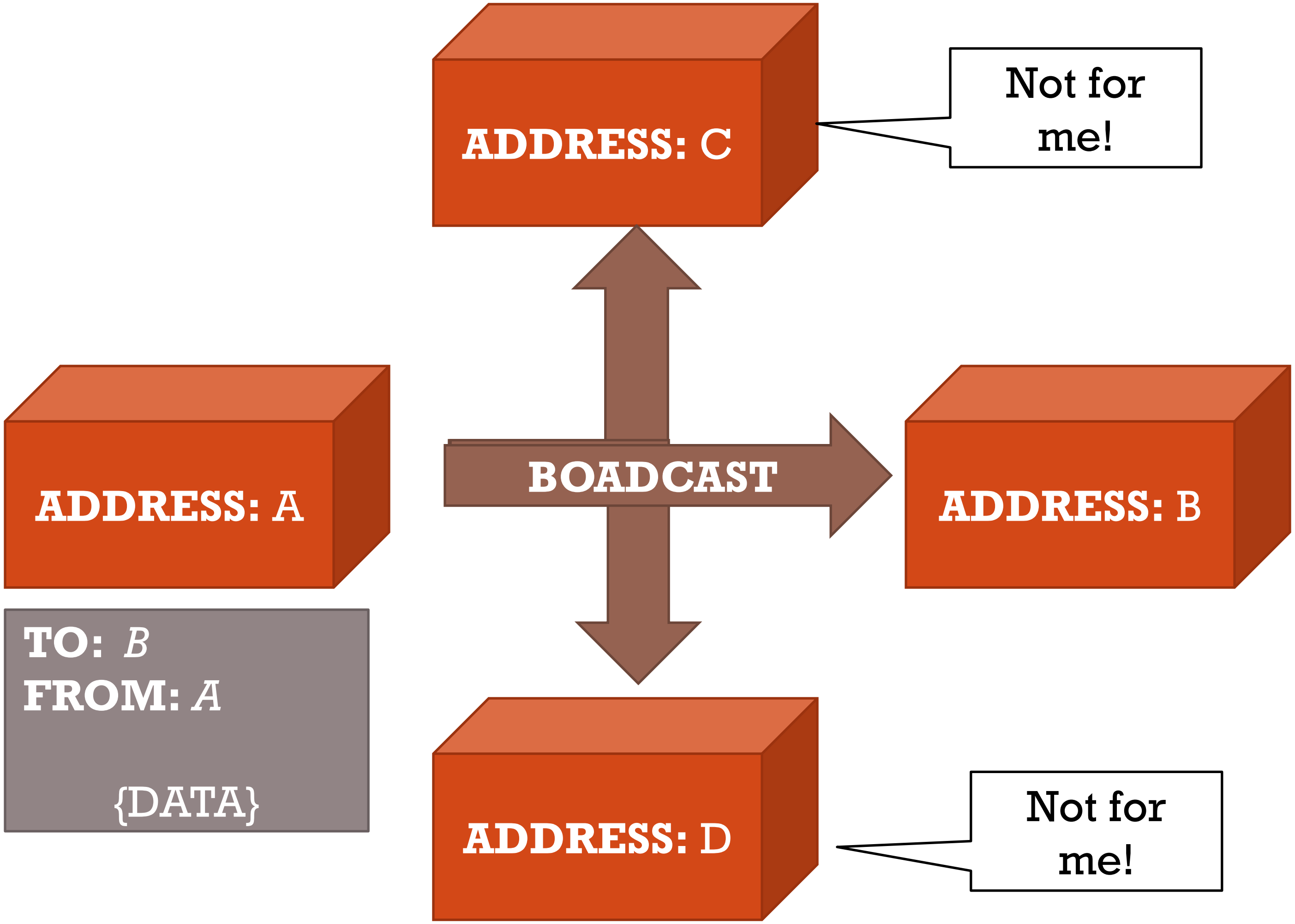
- At the MAC layer, protocol connects 2 endpoints. Typically:
  - Has its own addressing scheme (MAC address)
  - Controls who talks when
  - Provides error detection and *error correction*
- IP (Internetwork Protocol)
  - Connects many different networks of different media types
  - Global addressing scheme
- TCP
  - Reliable, in-order delivery (Session)
  - Multiplexing

LOCAL NETWORK CONCEPTS

- Local Area Network (LAN) – Direct node-to-node connection
- Usually involves a “medium”, hence Medium Access Control
- Nodes can send or receive MAC packets w/ MAC addresses
- ***Broadcast*** typically used for discovery

"OLD" ETHERNET (< 10MBPS)

- Broadcast used for ***everything***
- Each node would only respond to packets addressed to it
- Upon detecting a collision, node would randomly back-off



ADDRESS: C

Not for me!

ADDRESS: A

BOADCAST

ADDRESS: B

TO: *B*
FROM: *A*

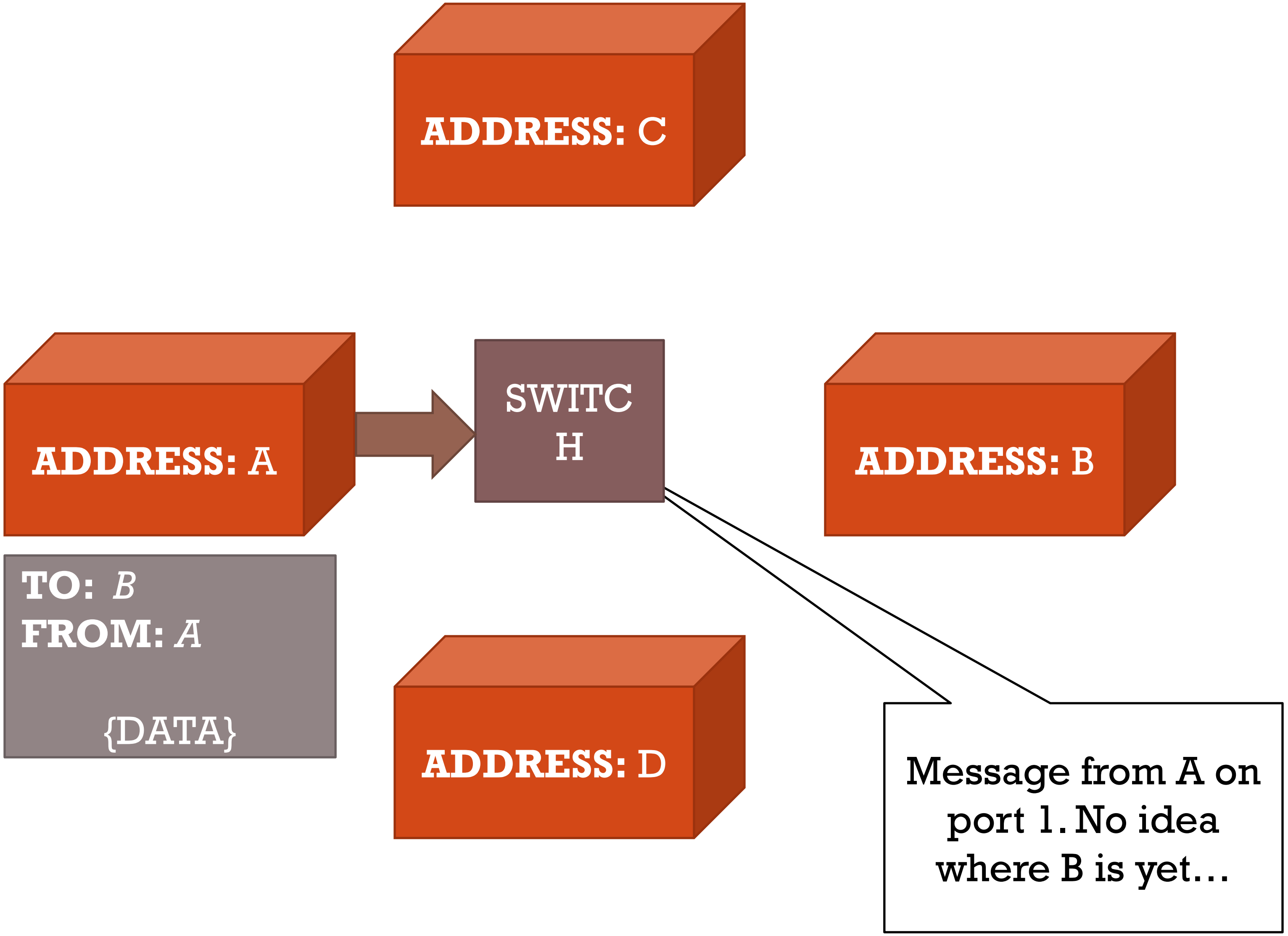
{DATA}

ADDRESS: D

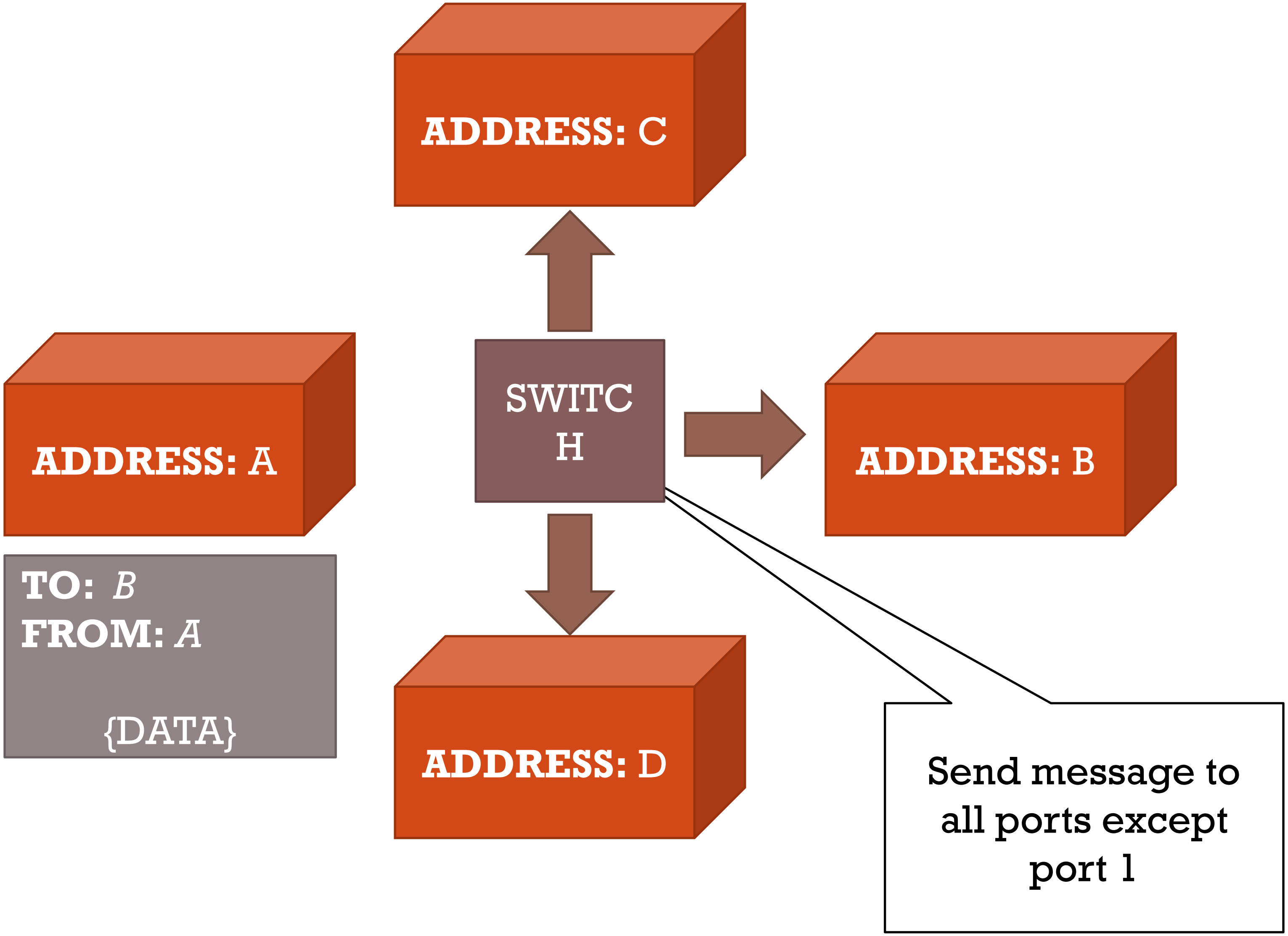
Not for me!

MODERN ETHERNET

- Switches ensure the data is delivered to the right node
- Broadcast (flooding) used to find a node the first time
- Much more efficient; enables 100Mbs+



Message from A on port 1. No idea where B is yet…



ADDRESS: C

ADDRESS: A

SWITC
H

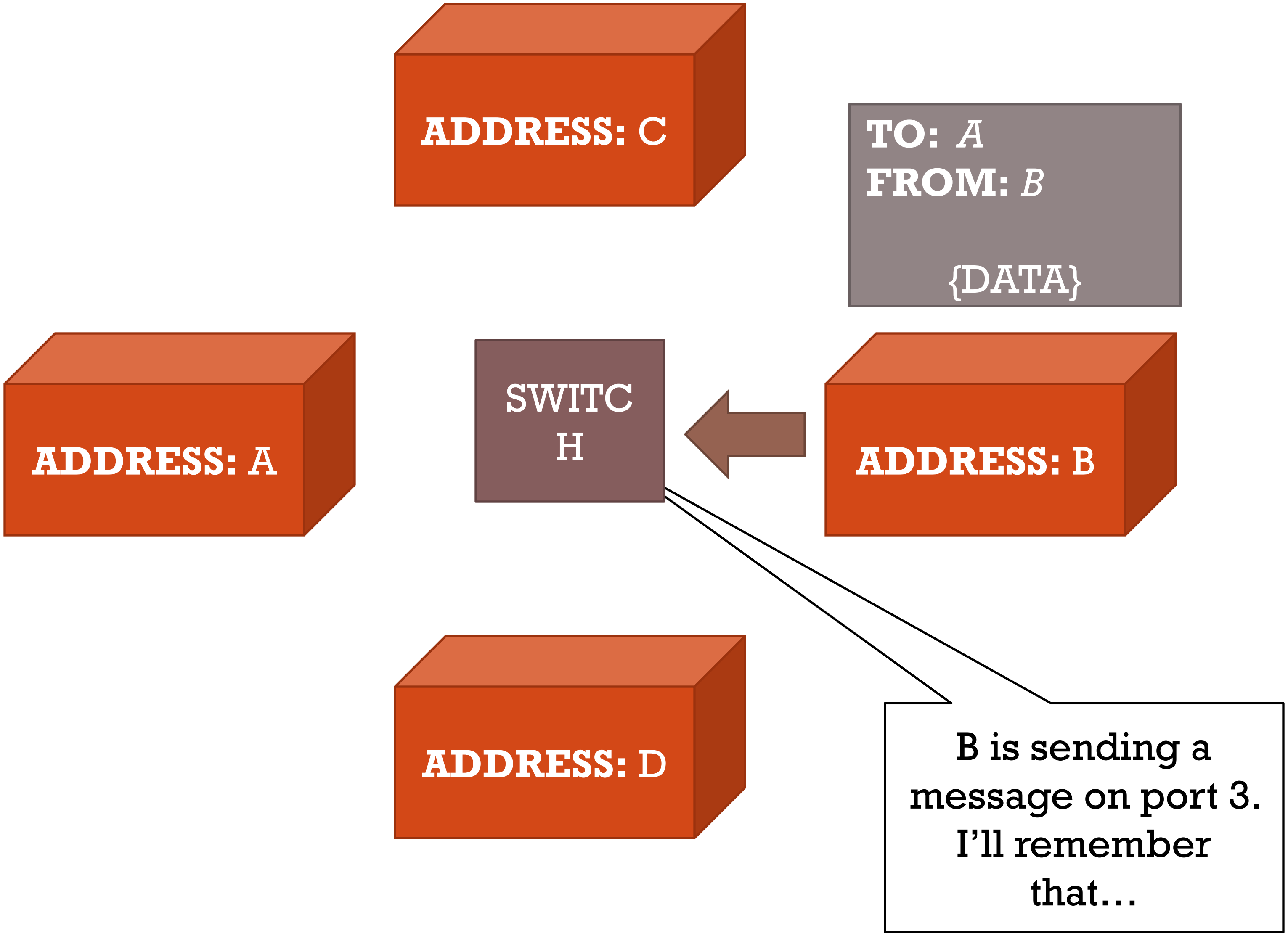
ADDRESS: B

TO: *B*
FROM: *A*

{DATA}

ADDRESS: D

Send message to
all ports except
port 1



ADDRESS: C

TO: *A*
FROM: *B*

{DATA}

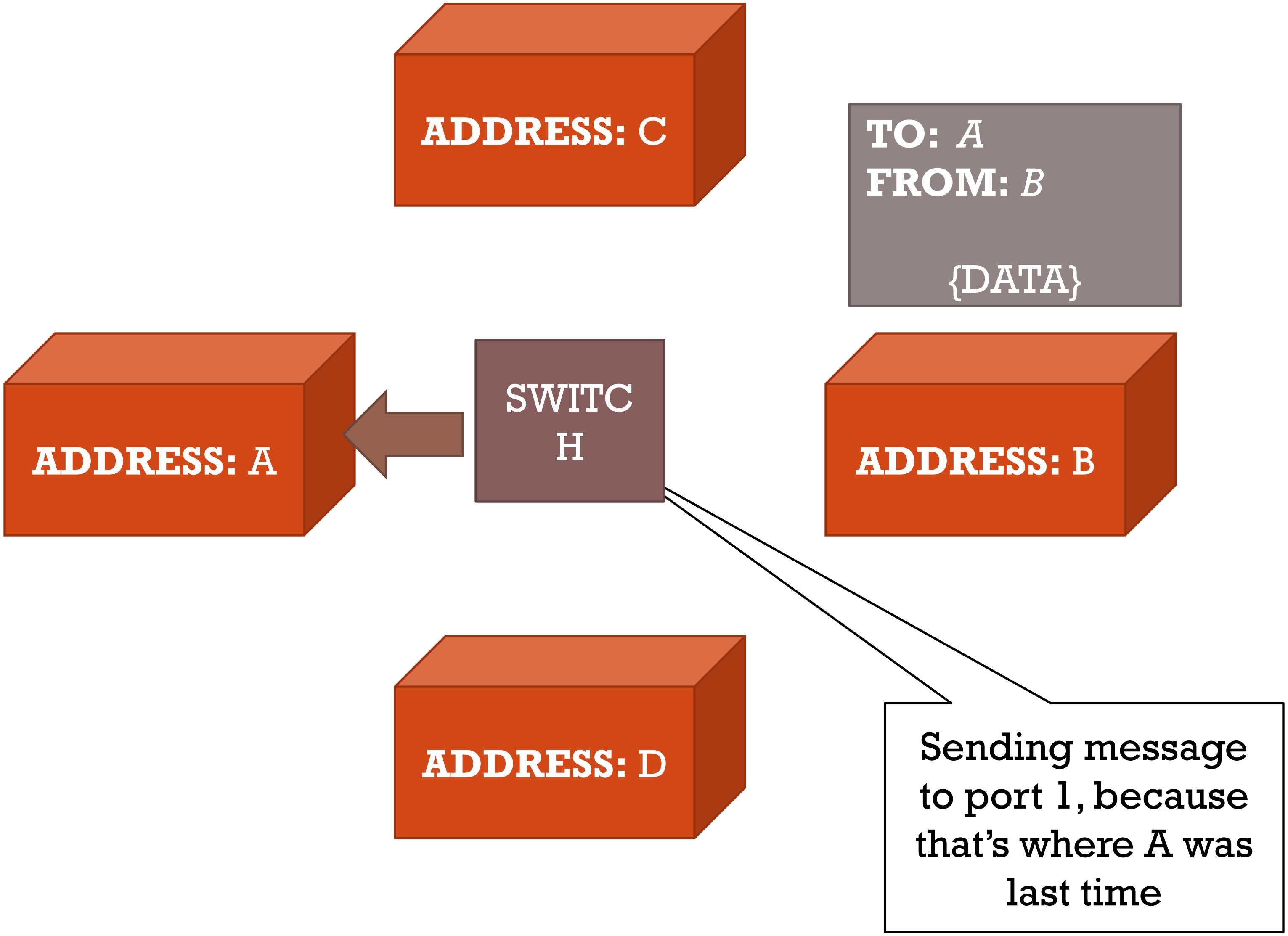
ADDRESS: A

SWITC
H

ADDRESS: B

ADDRESS: D

B is sending a
message on port 3.
I’ll remember
that…



ADDRESS: C

TO: *A*
FROM: *B*
{DATA}

ADDRESS: A

SWITC
H

ADDRESS: B

ADDRESS: D

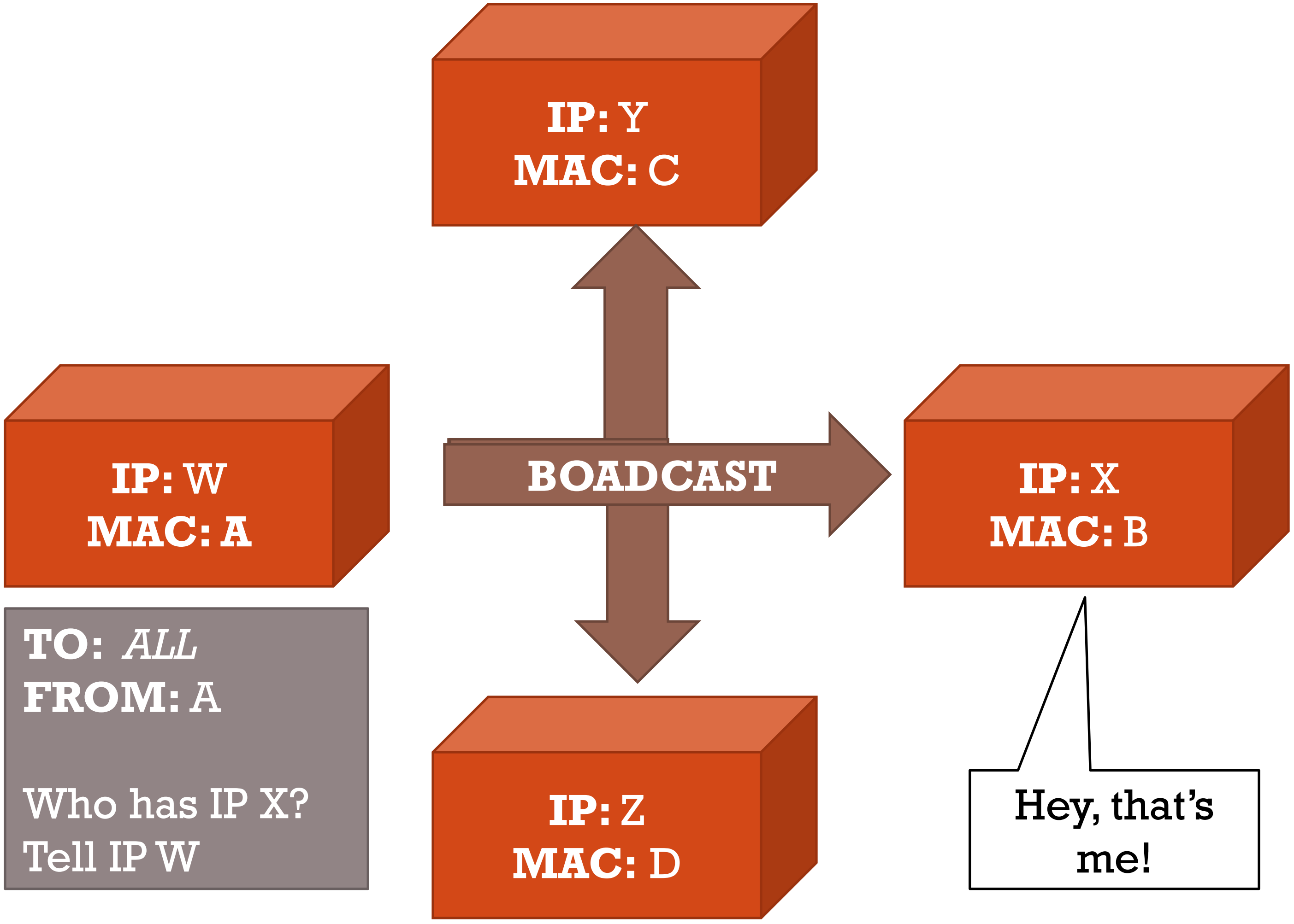
Sending message to port 1, because that's where A was last time

IP ADDRESS MAPPING

- MAC addresses are great, but app layer doesn’t use them
- Even when communicating on your LAN, you use IP addrs
- To communicate, IP-to-MAC address mapping required
- Enter the Address Resolution Protocol (ARP)

ARP BROADCAST DISCOVERY

- Every LAN has an explicit broadcast mechanism
- Ethernet typically uses broadcast address ***FF:FF:FF:FF:FF:FF***
- When IP with unknown MAC mapping needed, broadcast
- Ask all nodes on the LAN "who has IP addr X, tell IP addr W"



IP: Y
MAC: C

IP: W
MAC: A

BOADCAST

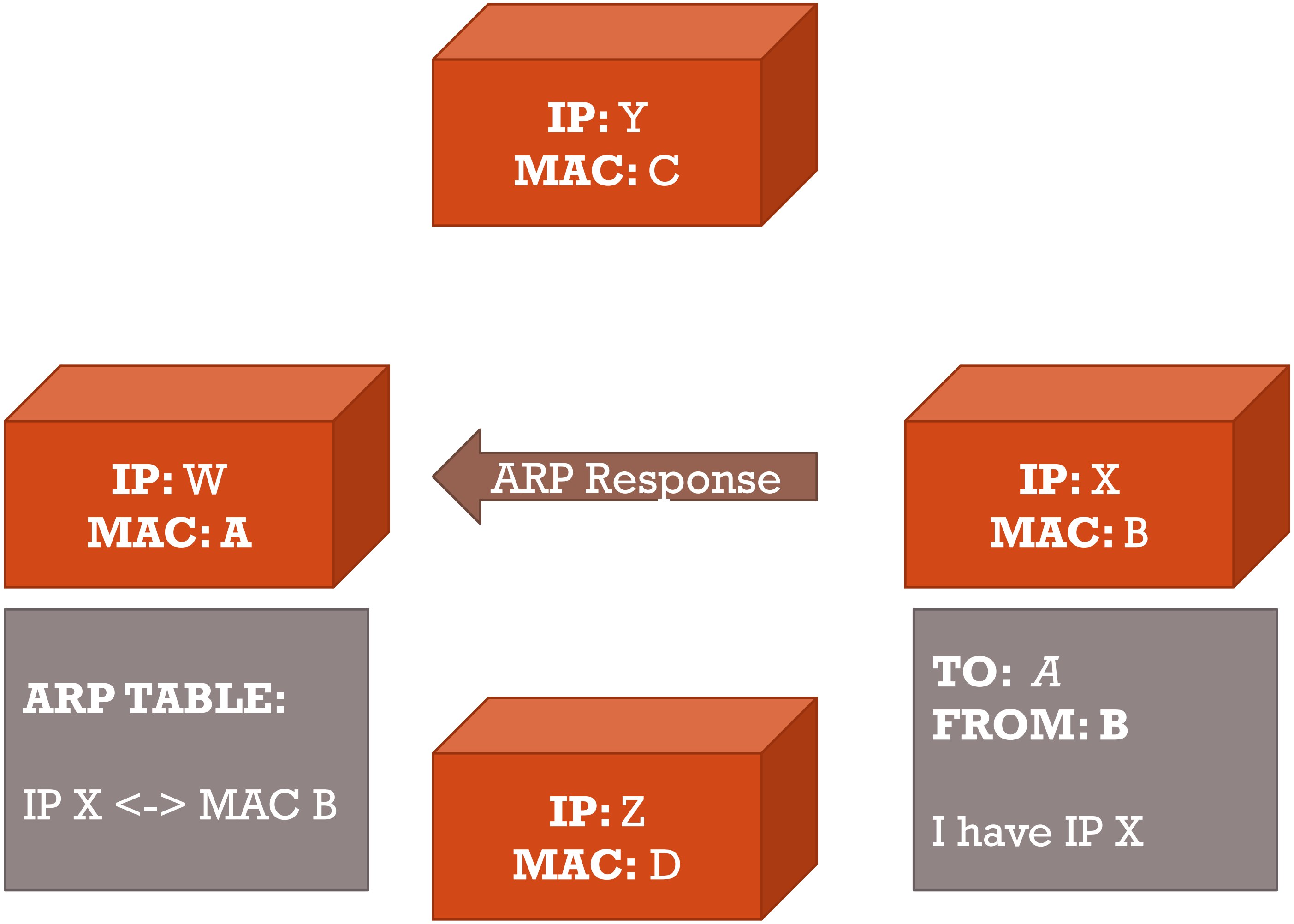
IP: X
MAC: B

IP: Z
MAC: D

TO: *ALL*
FROM: A

Who has IP X?
Tell IP W

Hey, that's
me!



IP: Y
MAC: C

IP: W
MAC: A

ARP Response

IP: X
MAC: B

ARP TABLE:

IP X <-> MAC B

IP: Z
MAC: D

TO: *A*
FROM: B

I have IP X

IP TO MAC CONVERSION

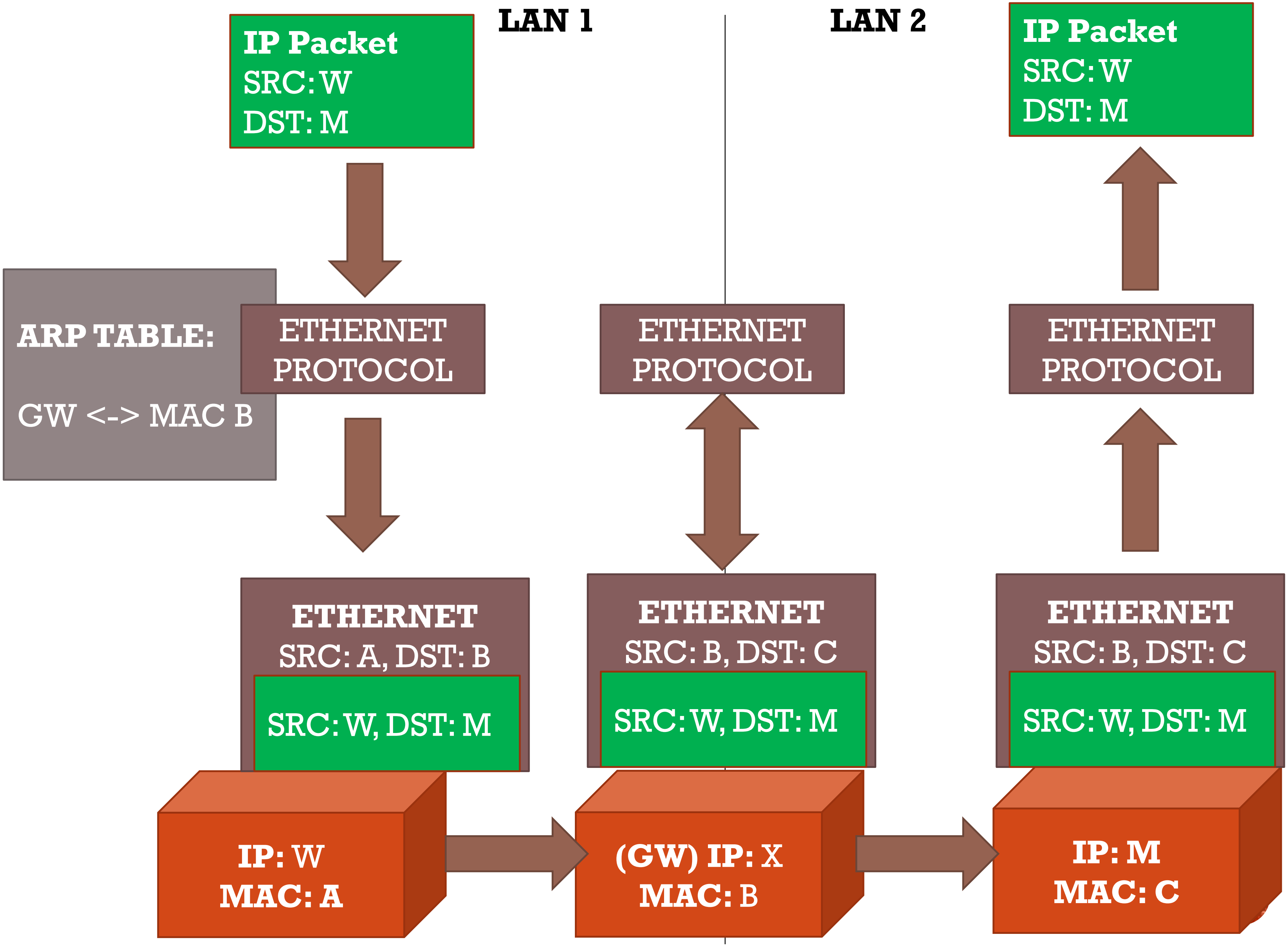
- Once ARP has the IP to MAC in the table, conversion is easy
- Translation occurs in Ethernet layer during “push” from IP
- IP destination addr is looked up in ARP table
- PS, what “layer” in OSI is ARP?

INTERNETWORKING CONCEPTS

- We've covered how to talk to a LAN node.
- What about a node outside our broadcast domain?
- Requires ***internetwork routing***

EXITING THE LAN

- A ***gateway*** node is connected to TWO+ LANs
- For any external IP address, use the MAC addr of the gateway
- (True dest IP addr remains the same!)
- Gateway node will examine the dest IP addr
- Gateway node determines where to send the data next



LAN 1

LAN 2

IP Packet
SRC: W
DST: M

ARP TABLE:

GW <-> MAC B

ETHERNET PROTOCOL

ETHERNET
SRC: A, DST: B
SRC: W, DST: M

IP: W
MAC: A

ETHERNET PROTOCOL

ETHERNET
SRC: B, DST: C
SRC: W, DST: M

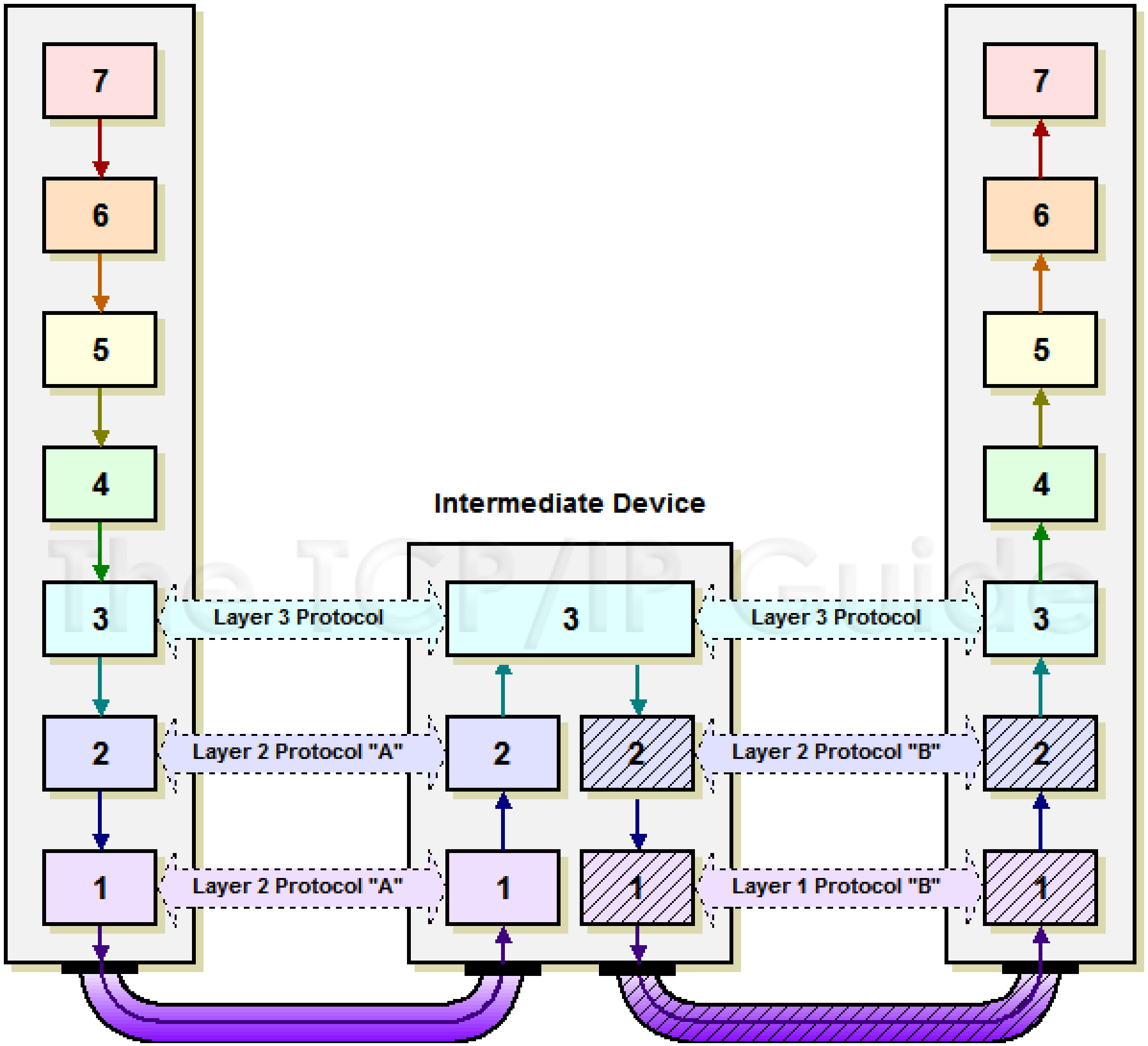
(GW) IP: X
MAC: B

IP Packet
SRC: W
DST: M

ETHERNET PROTOCOL

ETHERNET
SRC: B, DST: C
SRC: W, DST: M

IP: M
MAC: C



http://www.tcpipguide.com/free/t_IndirectDeviceConnectionandMessageRouting.htm

ROUTING ACROSS THE INTERNET

- Data may have to move up a hierarchy
- You to a Tier-3 ISP to a Tier-2 ISP, to a Tier 1 ISP
- Top level ISP’s usually manage the “Autonomous Systems” (AS)
- AS’s are the top level of the Internet in terms of routing
- AS’s are interconnected and represent the “backbone”

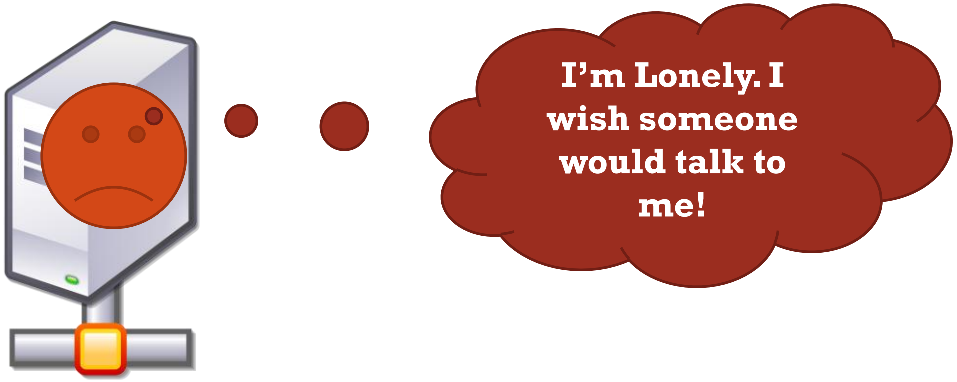
DOMAIN NAME SYSTEM (DNS)

- We don’t usually browse the web with IP addresses
- You can, of course. Try browsing to 172.217.13.4
- But what a pain to remember
- DNS is how we convert “google.com” to “172.217.13.4”

CLIENT-SERVER CONCEPTS

- Put resources in a high-performance, centralized machine
- Clients can be much “dumber” *by comparison*
- Much more efficient
  - Sharing data between devices, applications, and people
  - Access from multiple locations (including hackers!)
  - Time-sharing a central machine is more scalable & cost-effective

SERVER ABSTRACTION

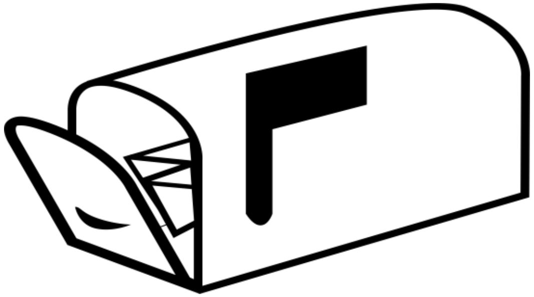


I’m Lonely. I wish someone would talk to me!

This Photo by Unknown Author is licensed under CC BY-SA

SERVER ***LISTENS*** FOR INCOMING REQUESTS

PREVIEW OF TCP/IP



This Photo by Unknown Author is licensed under CC BY-NC

This Photo by Unknown Author is licensed under CC BY-SA

Now I have an Address/Port! Maybe I'll get Requests!

SERVER HAS AN **IP ADDRESS** AND **TCP PORT**

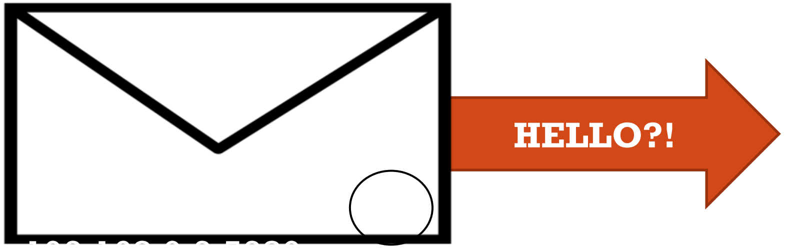
MEANWHILE, CLIENT ABSTRACTION

This Photo by Unknown Author is licensed under CC BY-NC

CLIENT ***CONNECTS*** TO MAKE OUTBOUND REQUESTS

TCP/IP AGAIN

[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

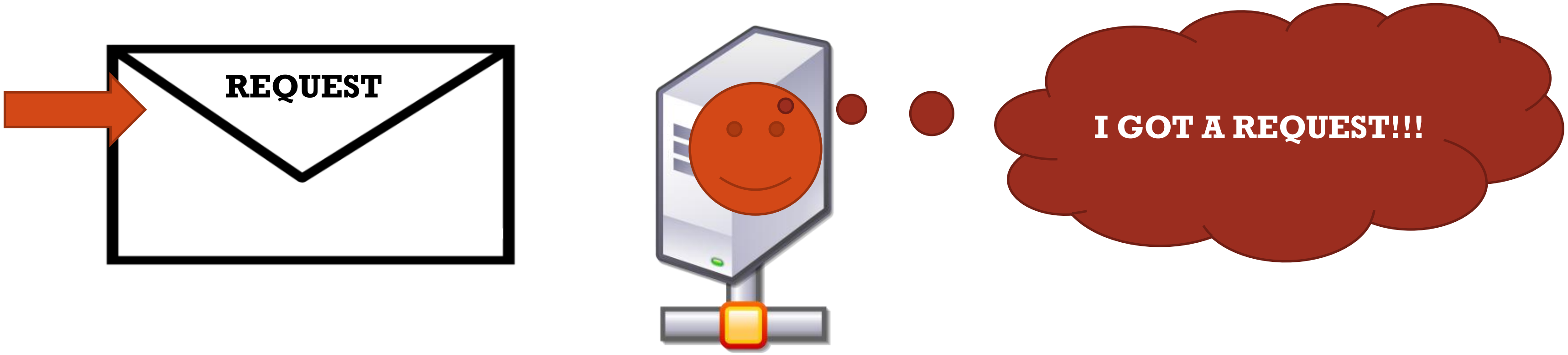


[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

Usually random

CLIENT ***CONNECTS*** TO MAKE OUTBOUND REQUESTS

INCOMING REQUEST



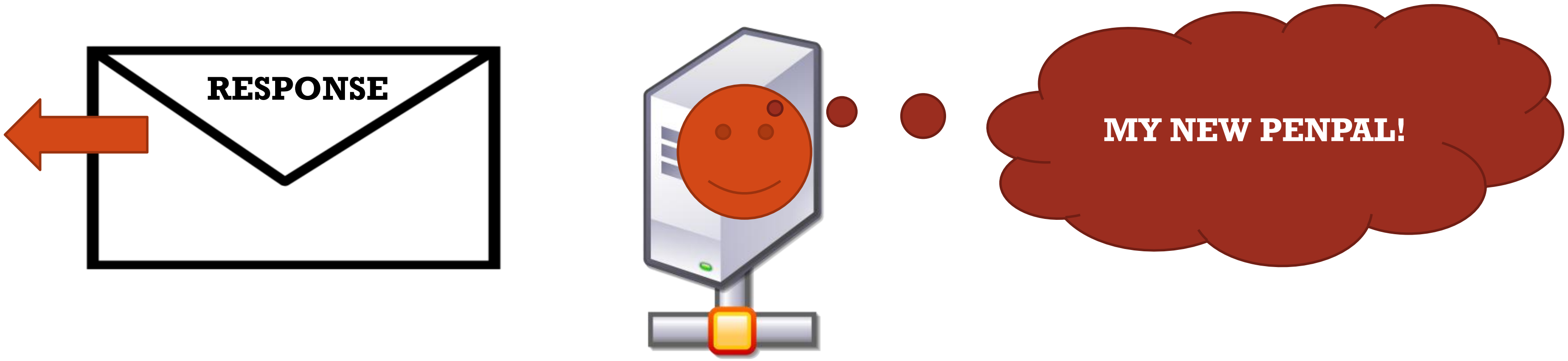
REQUEST

I GOT A REQUEST!!!

This Photo by Unknown Author is licensed under CC BY-SA

SERVER RECEIVES REQUEST

REQUEST RESPONSE



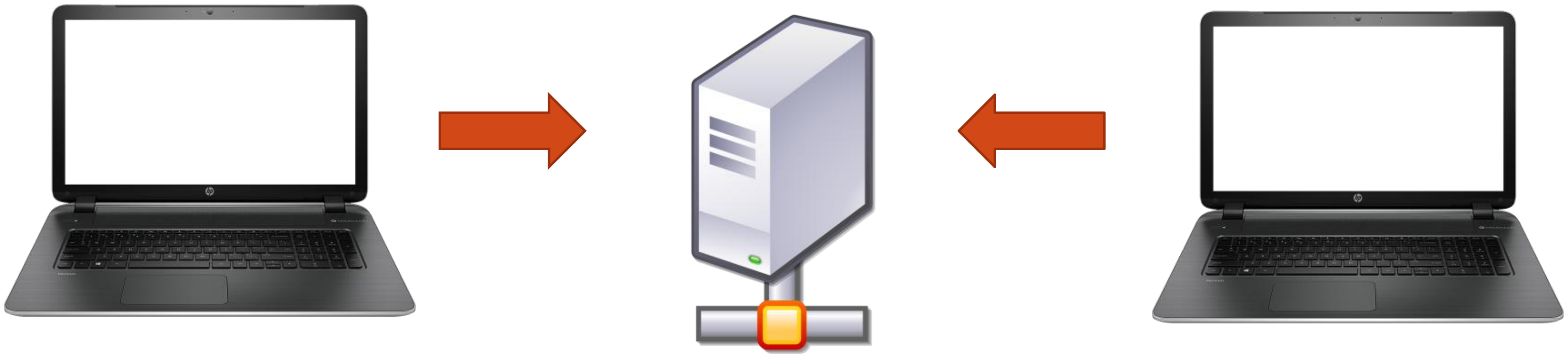
RESPONSE

MY NEW PENPAL!

This Photo by Unknown Author is licensed under CC BY-SA

SERVER **INVERTS TO/FROM** FOR RESPONSE

SERVER LISTENS TO MANY REQUESTS



SERVER USES **(SRC IP, SRC PORT, DST IP, DST PORT)*** TO MULTIPLEX

This is how one server on one port (e.g., webserver) handles many clients, even from the same computer

192.168.1.1, 5280, 192.168.1.2, 80	**Web Server Process 1**
192.168.1.1, 9019, 192.168.1.2, 80	Web Server Process 2