Networks, Security, and Privacy

158.235

A/ ard

Massey Uni

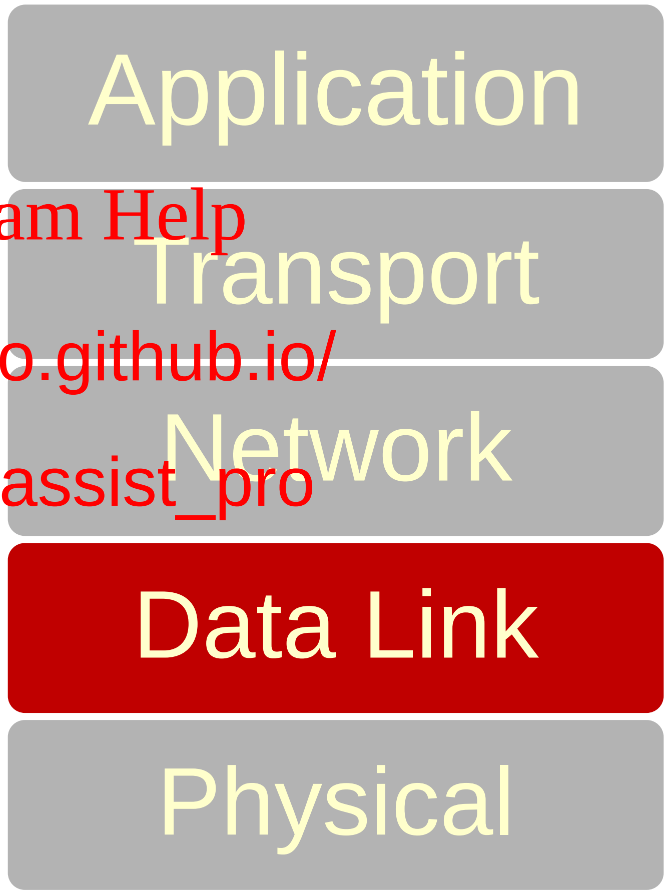
Lay

Reading: Chapter 4 in the prescribed textbook

Introduction

- Layer 2 in the Internet model
- Responsible for moving messages (datagram) from one device (another phy node over a
- Major functions of a data link layer protocol
  - Error Control
  - Flow Control
  - Link layer addressing

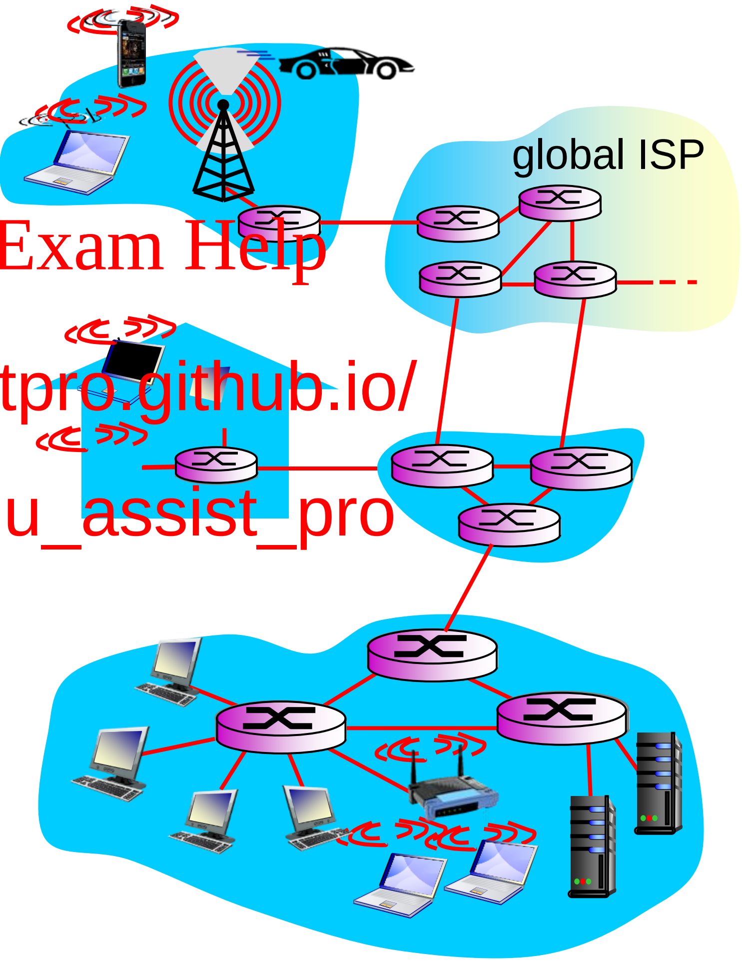
Internet Model



Introduction

terminology:

- **hosts and routers: nodes**
- **communication channels that connect adjacent nodes along communication**
  - **wired links**
  - **wireless links**
  - **LANs**
- **layer-2 packet: frame, encapsulates datagram**



Introduction

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame r intermediat 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide reliable data transfer over link

transportation analogy:

- trip from Palmerston North to Disney Land, LA
  - – Taxi: City center to PN airport
  - : PN to Auckland
  - : Auckland to LAX
  - X to Disney Land
- – **datagram**
- transport segment = **communication link**
- transportation mode = **link layer protocol**
- travel agent = **routing algorithm**

Link layer services

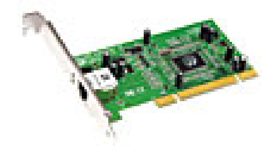
- ***framing, link access:***
  - •encapsulate datagram into frame, adding header, trailer
  - •“MAC” addresses used in frame headers to identify sour IP address!)

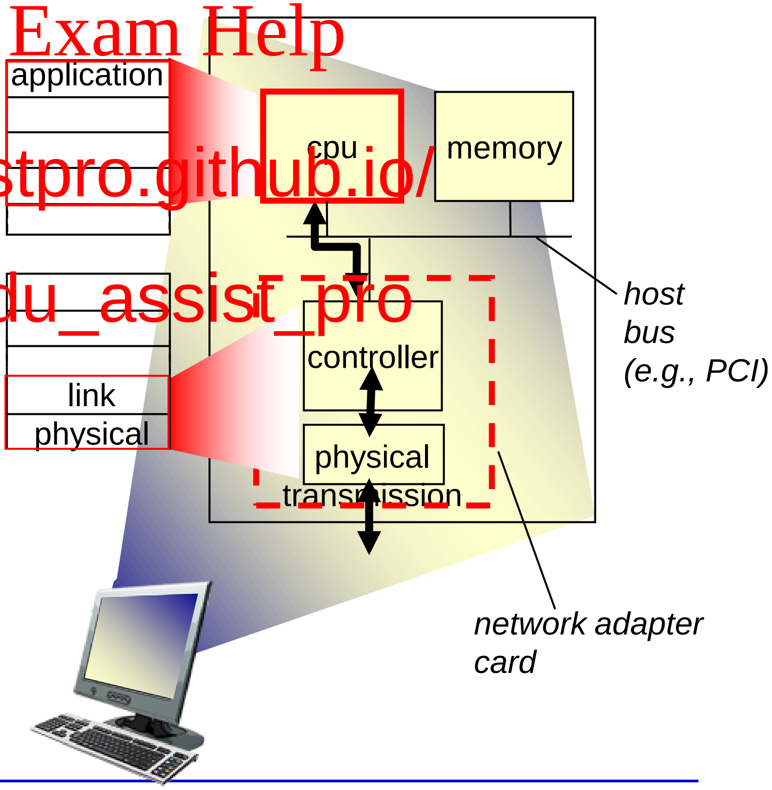
- ***flow control***
  - – pacing between adjacent sen eiving nodes

- ***error detection*:**
  - – errors caused by signal attenuation, noise.
  - – receiver detects presence of errors:
- ***error correction:***
  - – receiver identifies ***and corrects*** bit error(s) without resorting to retransmission

Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet car
    card; Ethern
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware





Data Link Layer

- **Error Control**

- **Flow**

- **Link Addre**

Error Control

- Network errors
  - Types
    - Corrupted data
    - Lost data
  - Caused b ssion (not humans)
- Networks should be d ith:
  - Error prevention
  - Error detection
  - Error correction

Sources of Network Errors

- **Line noise and distortion**
  - **Major reason for errors and caused by several sources**
  - **More likely on electrical media and lower-end cables (e.**
  - **Undesira**
  - **Degrades uit**
  - **Manifestation**
    - Extra bits
    - Flipped bits
    - Missing bits

Sources of Errors and Prevention

Source of Error	What Causes It	How to Prevent or Fix
White Noise	Movement of electrons	Increase signal strength
Impulse Noise	Sudden increases in electricity (e.g., lightning)	Shield or move the wires
Cross-talk	M wi	Increase the guardbands or move or shield the wires
Echo	Poor (misaligned) conne	ix the connections or tune quipment
Attenuation	Gradual decrease in signal over distance	Use repeaters
Intermodulation noise	Signals from several circuits combine	Move or shield the wires

Error Detection

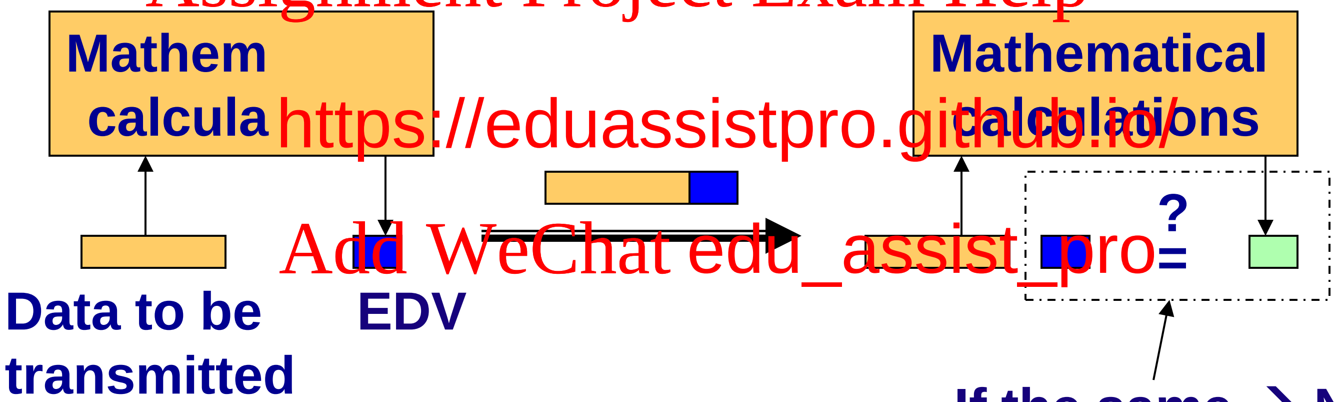
- Receivers need to know when the data transmitted is not correct
- Add “check value” (error detection value) to messag

- Check value produced by mathematical formula

Error Detection

Sender calculates an Error Detection Value (EDV) and transmits it along with data

Receiver recalculates EDV and checks it against the received EDV



– If the same → No errors in transmission

– If different → Error(s) in transmission

Error Detection Techniques

- Parity checks
- Checksum
- Cyclic Redundancy Check (CRC)

Parity Checking

- One of the oldest and simplest
- A single bit added to each character
  - Even parity: number of 1's remains even
  - Odd parity: number of 1's remains odd

- Receiving e bit

  - If one bit h r the received parity bit will differ from the d one

- Simple, but doesn't catch all errors
  - If two (or an even number of) bits have been transmitted in error at the same time, the parity check appears to be correct
  - Detects about 50% of errors

Examples of Using Parity

To be sent: Letter V in 7-bit ASCII: 0110101

EVEN parity

sender

receiver

Add a bit so tha number of all transmitted 1's EVEN

ODD parity

sender

receiver

01101011

Add a bit so that the number of all transmitted 1's is ODD

parity

Checksum

- A checksum (usually 1 byte) is added to the end of the message
- It is 95% effective

- Method:

  - Add decimal values of eac n the message
  - Divide the sum by 255
  - The remainder is the checksum value

CRC

- Cyclic redundancy check (CRC)
  - Treats message as a single binary number
  - Divides by a preset number
  - Uses rem lue
- Preset nu hat remainder is the corre r of bits
- Modes:
  - CRC-16 (~99.998% error detection rate)
  - CRC-32 (>99.99999% error detection rate)

Cyclic Redundancy Check (CRC)

$$P / G = Q + R / G$$

Example:
P = 58
G = 8
Q = 7
R = 2

Message (treated as one long binary number)

Quotient

Remainder:
dded to the
essage as EDV
ld be 8 bits, 16
, 24 bits, or 32
long
–CRC16 has R of 16 bits

A fixed number (divisor) which determines the length of the R

- Most powerful and most common
- Detects 100% of errors (if number of errors <= size of R)
  - Otherwise: CRC-16 (99.998%) and CRC-32 (99.9999%)

Error Correction

- Once detected, the error must be corrected
- Error correction techniques
  - Retransmission (or, backward error correction)

    - Simple
    - Automa

    - This can also provide fl y limiting the number of messages se

  - Forward Error Correction
    - Receiving device can correct incoming messages without retransmission

Automatic Repeat reQuest (ARQ)

- Process of requesting a data transmission be resent
- Main ARQ protocols
  - Stop and Wait ARQ (A half duplex technique)
    - Sender s for acknowl next message
    - Receiver receives the m sends an acknowledgement, then next message
  - Continuous ARQ (A full duplex technique)
    - Sender continues sending packets without waiting for the receiver to acknowledge
    - Receiver continues receiving messages without acknowledging them right away

Stop and Wait ARQ

Sender | **Receiver**

Sends Packet A, then waits to hear from receiver.

Sends acknowledgement

Sends the next packet (B)

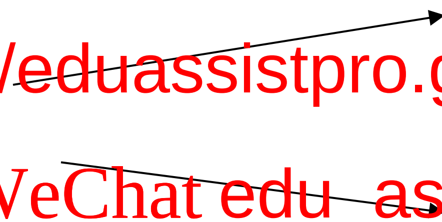
Sends negative acknowledgement

Resends the packet again

Sends acknowledgement

Continuous ARQ

Sender sends packets continuously without waiting for receiver to acknowledge



Not acknowledgme identify the packet being acknowledged.

Receiver sends back a NAK for a specific packet to be resent.

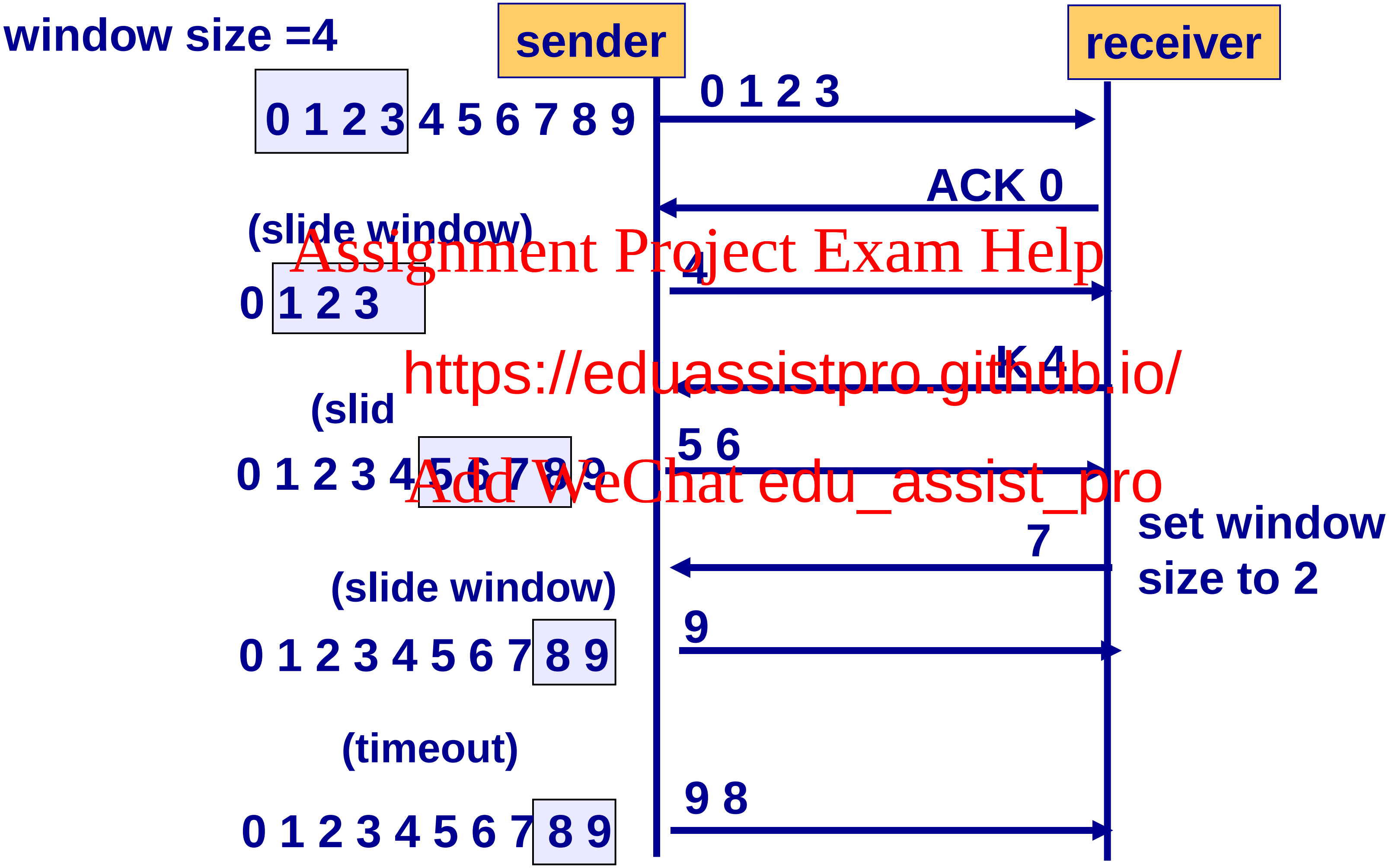
Data Link Layer

- Error Control
- **Flow**
- Link Addre

Flow Control with ARQ

- Ensuring that sender is not transmitting too quickly for the receiver
  - Stop-and-wait ARQ
    - Receiv AK when it is ready t ts
  - Continuous ARQ:
    - Both sides agree on the size of the "sliding window"
      - Number of messages that can be handled by the receiver without causing significant delays

Flow Control Example



Forward Error Correction

- Receiving device can correct incoming messages itself (without retransmission)
- Requires extra corrective information
  - Sent along with the data
  - Allows dat ted by the receiver
  - Amount of 50-100% of the data
- Used in the following situ
  - One way transmissions (retransmission not possible)
  - Transmission times are very long (satellite)
  - In this situation, relatively insignificant cost of FEC

Hamming Code – An FEC Example

- A scheme by adding parity bit intelligently such that *one* erroneous bit can be detected and corrected
- Bit position is split into 'parity bit' position and 'data bit' position:
  - parity bit oc 32, …

  - data bit occ s (3, 5, 6, 7, 9,…)
  - parity bit value calculation:
    - position 1 → check 1 bit, s so forth (1, 3, 5, …)
    - position 2 → check 2 bits, skip 2 bits (2, 3, 6, 7, 10, 11,…)
    - position 4 → check 4 bits, skip 4 bits (4-7, 12-15, …)

Hamming Code – Example

Data: 11011010

Even Parity

Position	1	2							9	10	11	12
Data	1	1							1	0	1	0
	P1	P2		P4								

P1: data at position 3, 5, 7, 9, 11 → 11111 (odd 1s) → Parity bit: 1

P2: data at position 3, 6, 7, 10, 11 → 10101 (odd 1s) → Parity bit: 1

P4: data at position 5, 6, 7, 12 →1010 (even 1s) → Parity bit: 0

P8: data at position 9, 10, 11, 12 → 1010 (even 1s) → Parity bit: 0

Data sent: 111010101010

Hamming Code – Example

Data Received: 111010101110

Position	1	2	3	4	5	6	7	8	9	10	11	12
Data	1	1	1	0	1	0	1	0	1	1	1	0

P1 P2

Check P1: data at p d 1s) → Parity bit: 1
- OK

Check P2: data at position 3, 6, 7, 10, n 1s) → Parity bit: 0
– Not OK

Check P4: data at position 5, 6, 7, 12 →1010 (even 1s) → Parity bit: 0
- OK

Check P8: data at position 9, 10, 11, 12 → 1110 (odd 1s) → Parity bit: 1
- Not OK

Parity bit at position 2 and 8 are incorrect.
The erroneous bit is placed at bit position 2+8 = 10

Data Link Layer

- Error Control
- Flow

- Link Addre

Address Resolution

- Addresses exist at different layers

Address Type	Example	Example Address
Application layer	Web address (URL)	www.indiana.edu
Network layer	I	9.79.78.193 (4 bytes)
Data link layer		6F-65-F8-33-8A (6 bytes)

- Addresses may be tra esolved) from one layer to another

Address Resolution

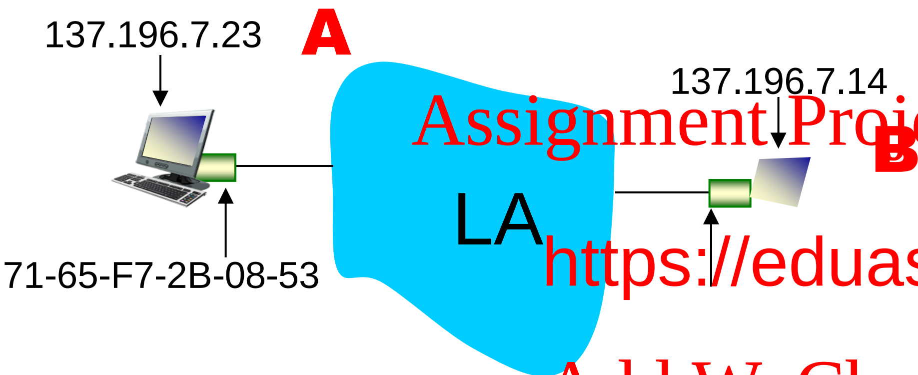
- **Data Link Layer Address Resolution**
  - **Identifying the MAC address of the next node (that packet must be forwarded)**
  - **Uses Address Resolution Protocol (ARP)**

ARP name resolution

- Identifying the MAC address by IP address
- Operation
  - Broadcast an ARP message to all nodes on a LAN asking which node has a certain IP address
  - Host with that IP address then responds by sending back its MAC addres
  - Store this M le
  - Send the me

ARP: same LAN

Question: how to determine a MAC address knowing its IP address?



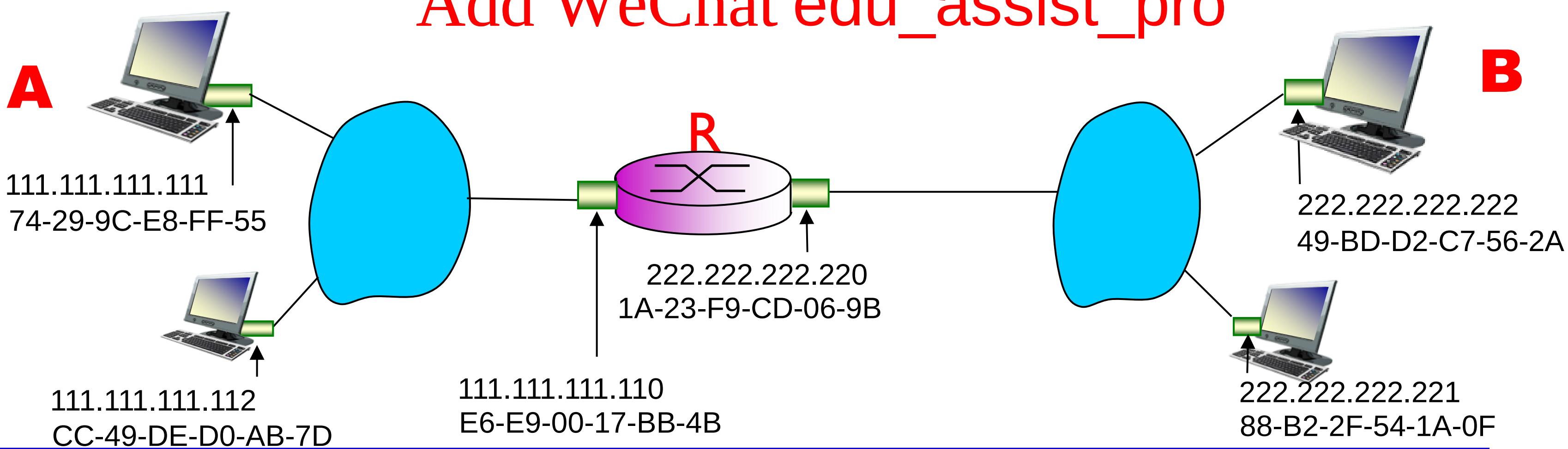
- A broadcasts ARP query packet, containing B's IP address
  - MAC address = FF-FF-FF-FF
  - s on LAN receive ARP query (broadcast)
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)

	ARP query	ARP reply
Src IP address	137.196.7.23	137.196.7.14
Dest IP address	137.196.7.14	137.196.7.23
Src MAC address	71-65-F7-2B-08-53	58-23-D7-FA-20-B0
Dest MAC address	FF-FF-FF-FF-FF-FF	71-65-F7-2B-08-53

Addressing: routing to another LAN

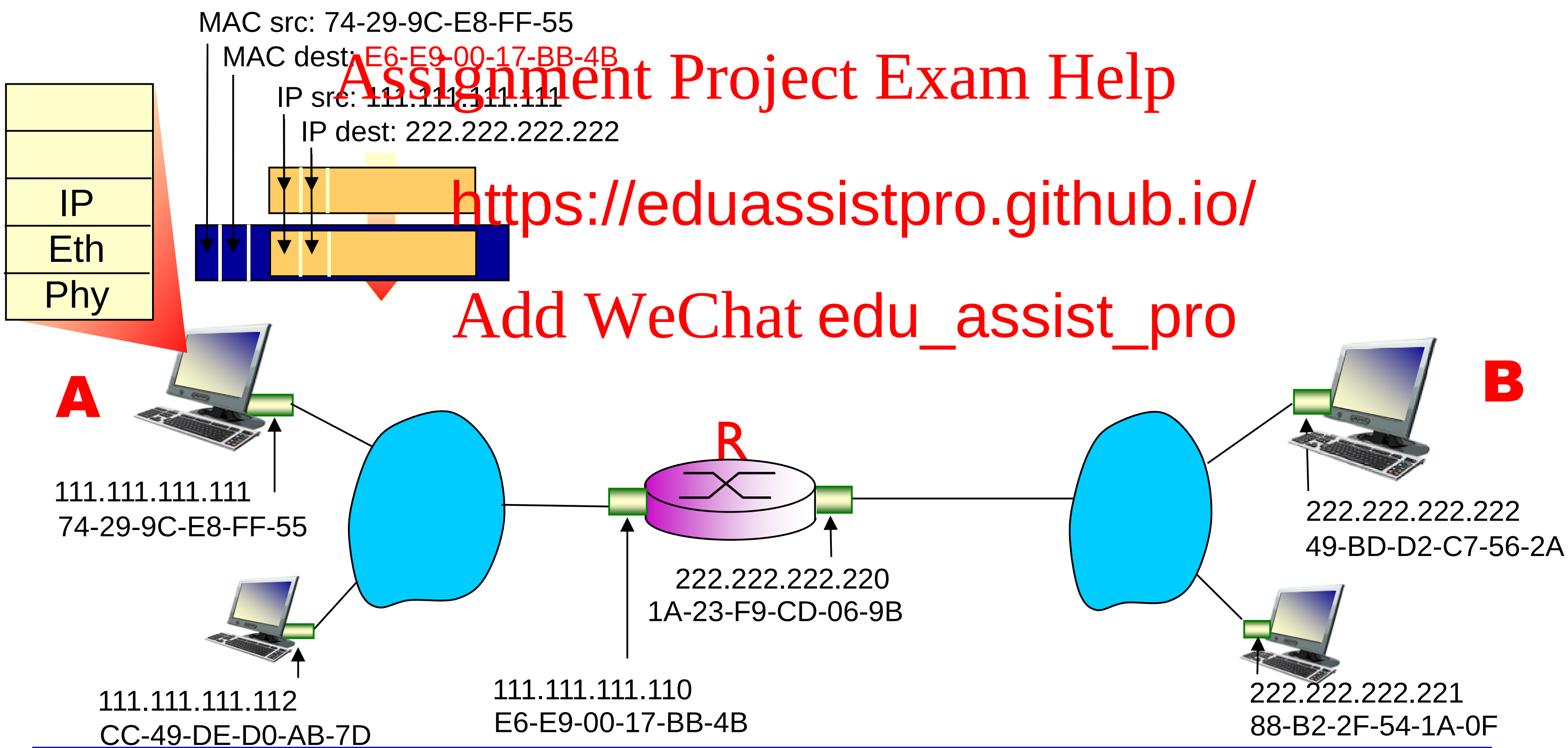
walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B's IP address
- assume A kn   hop router, R
- assume A kn



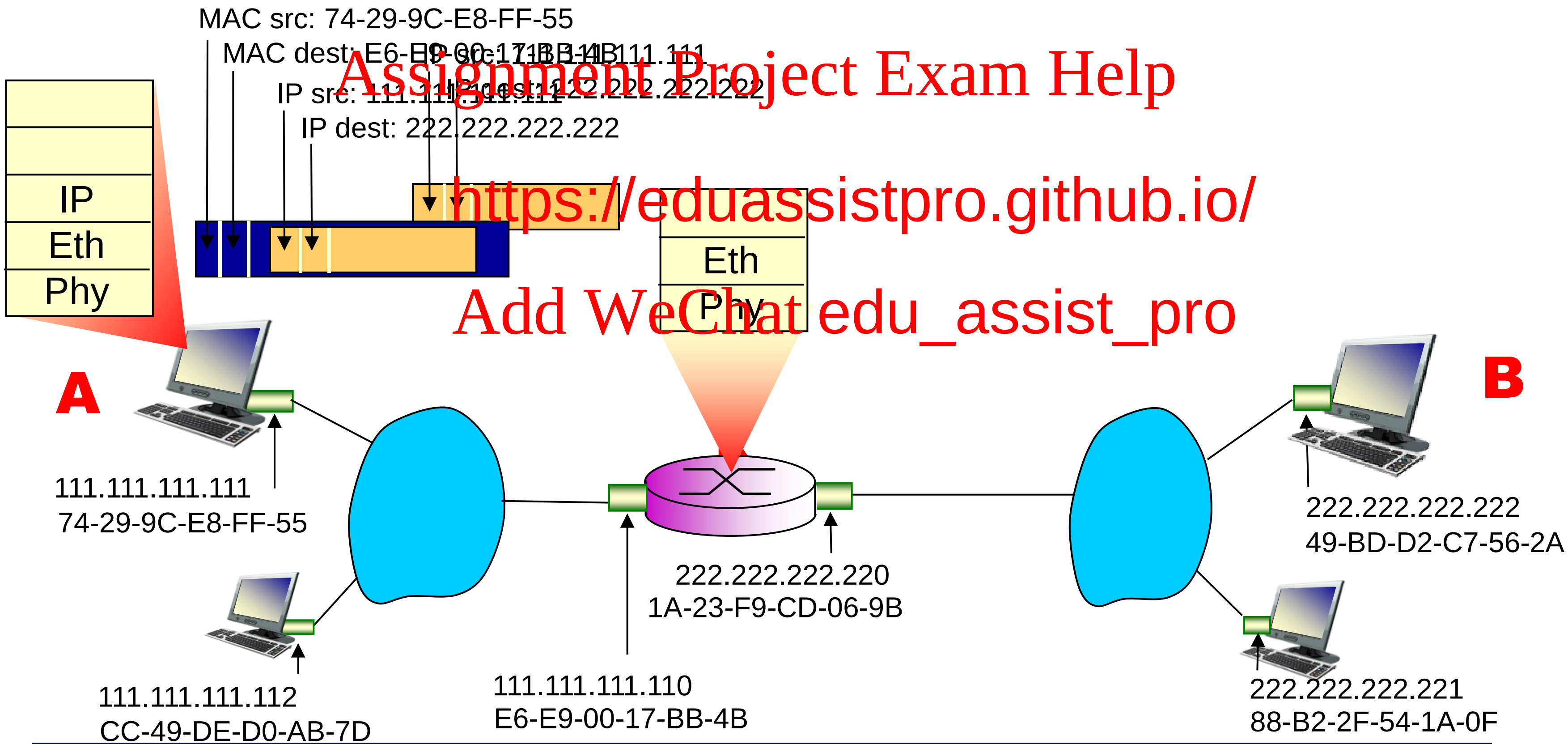
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



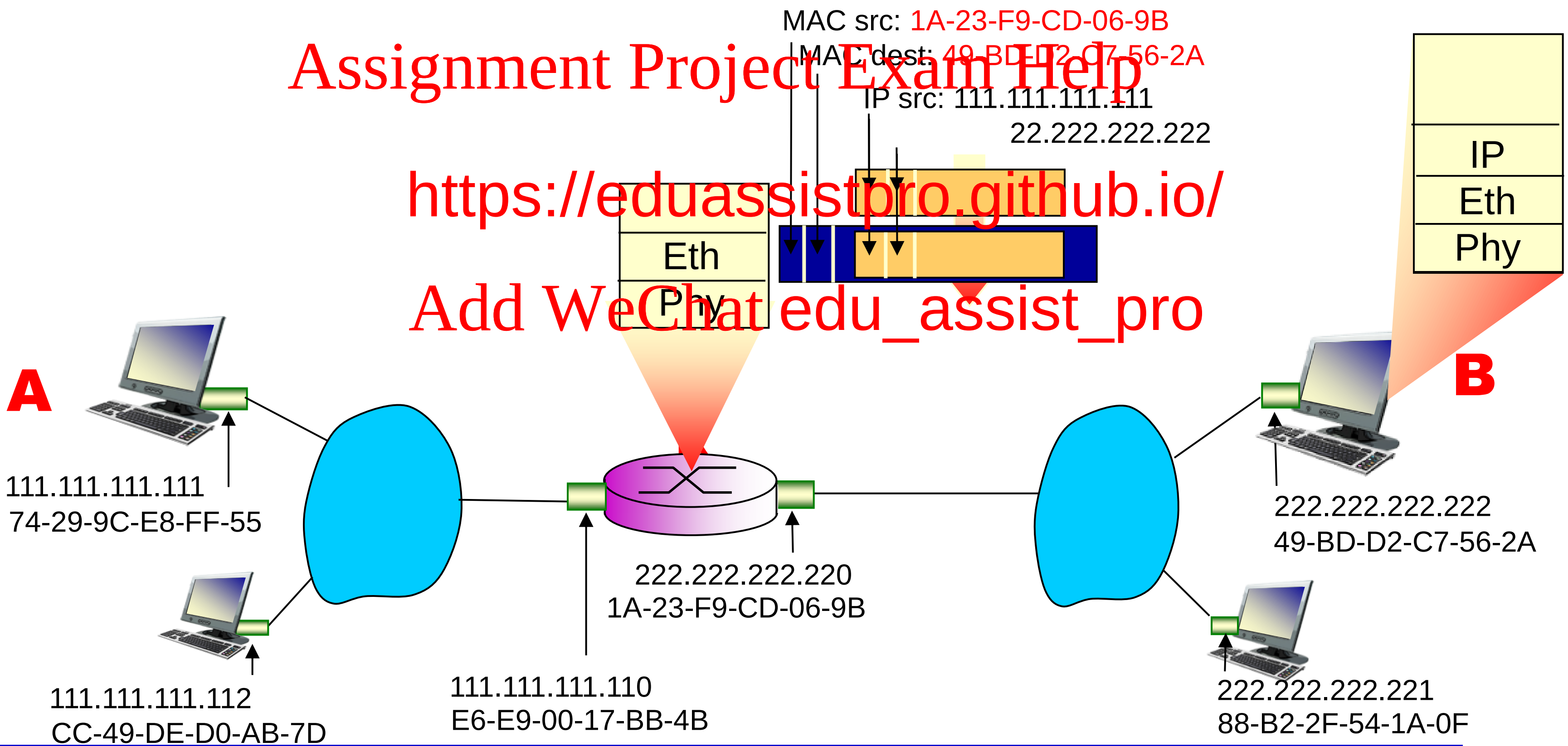
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



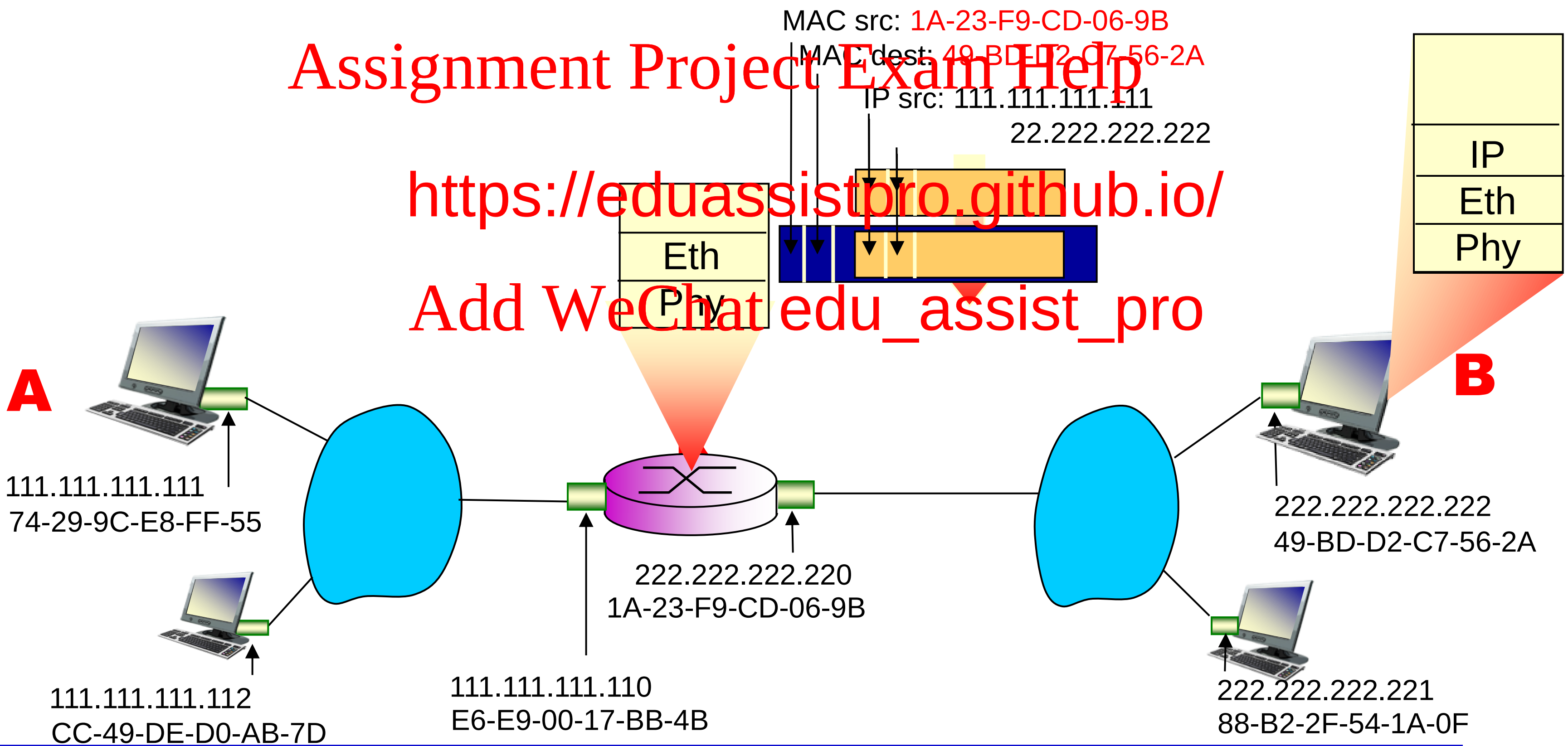
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



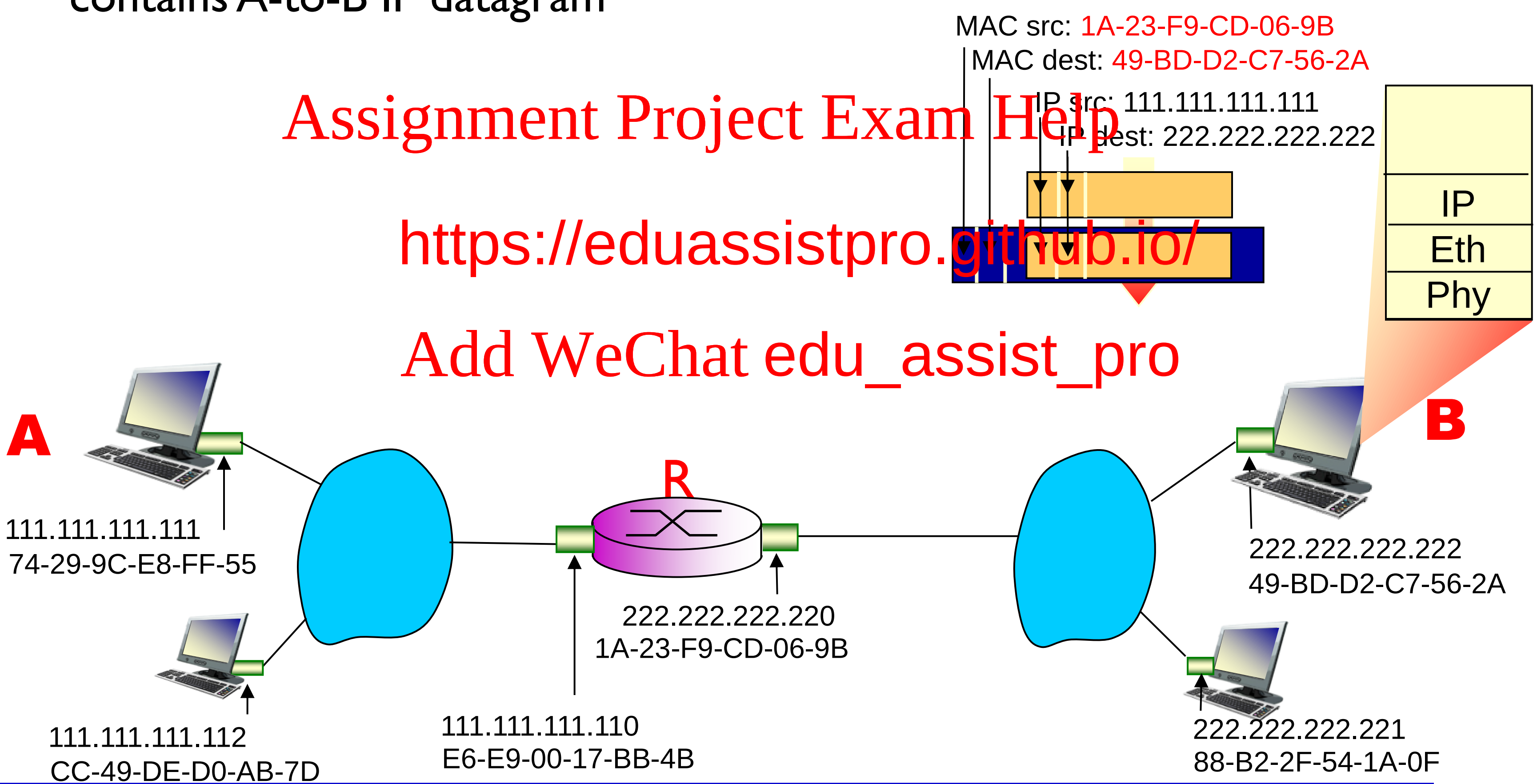
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Ethernet

"dominant" wired LAN technology:

- cheap $20 for NIC
- first widely used LAN technology

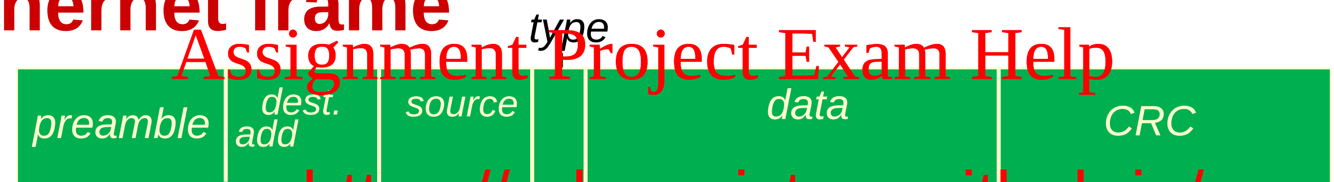
- simpler, ch and ATM

- kept up wit – 10 Gbps

Metcalfe's Ethernet sketch

Ethernet frame structure

sending adapter encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**



preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Ethernet frame structure

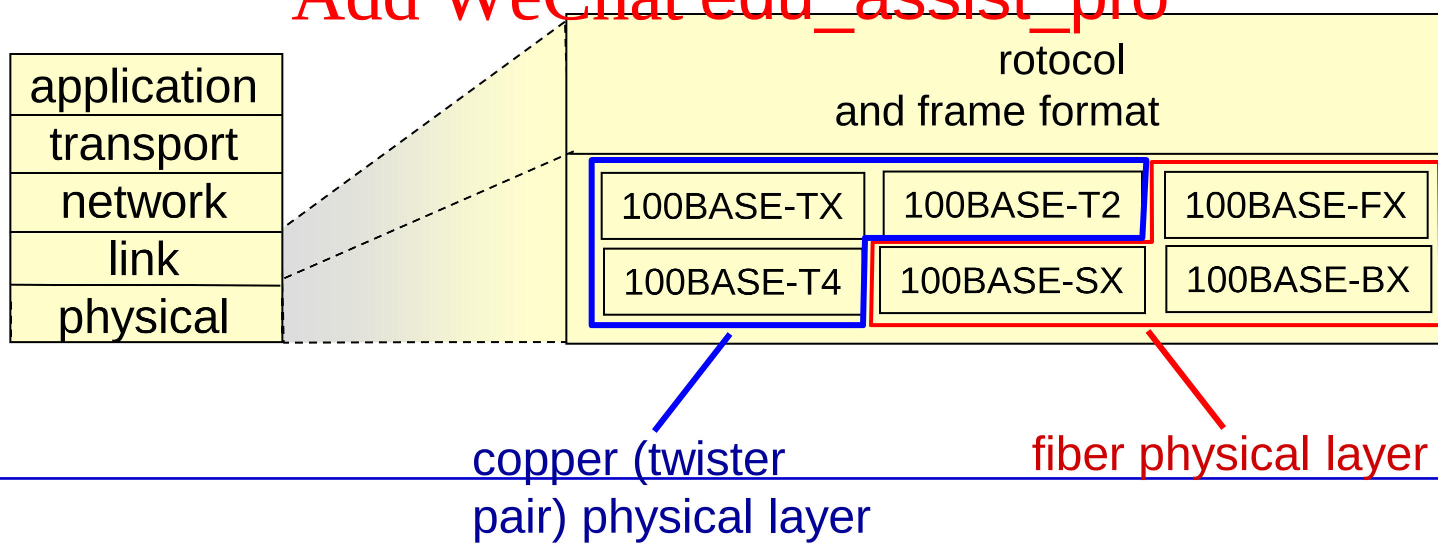
- *addresses:* 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes d ayer protocol
  - otherwise,

- *type:* indicates higher lay ol (mostly IP but others possible, e.g., X, AppleTalk)
- *CRC:* cyclic redundancy check at receiver
  - error detected: frame is dropped

preamble	dest. address	source address	type	data (payload)	CRC

802.3 Ethernet standards: link & physical layers

- ***many*** **different Ethernet standards**
  - **common MAC protocol and frame format**
  - **different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps**
  - **different p er, cable**

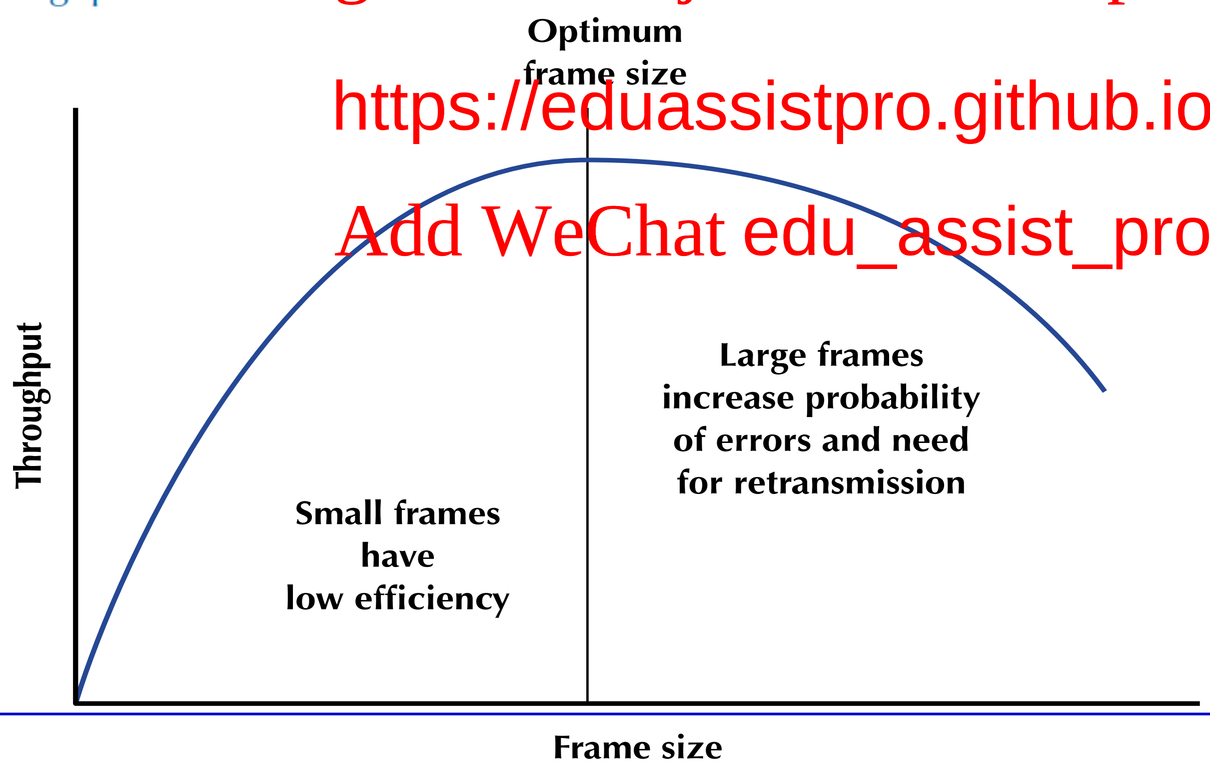


Transmission Efficiency

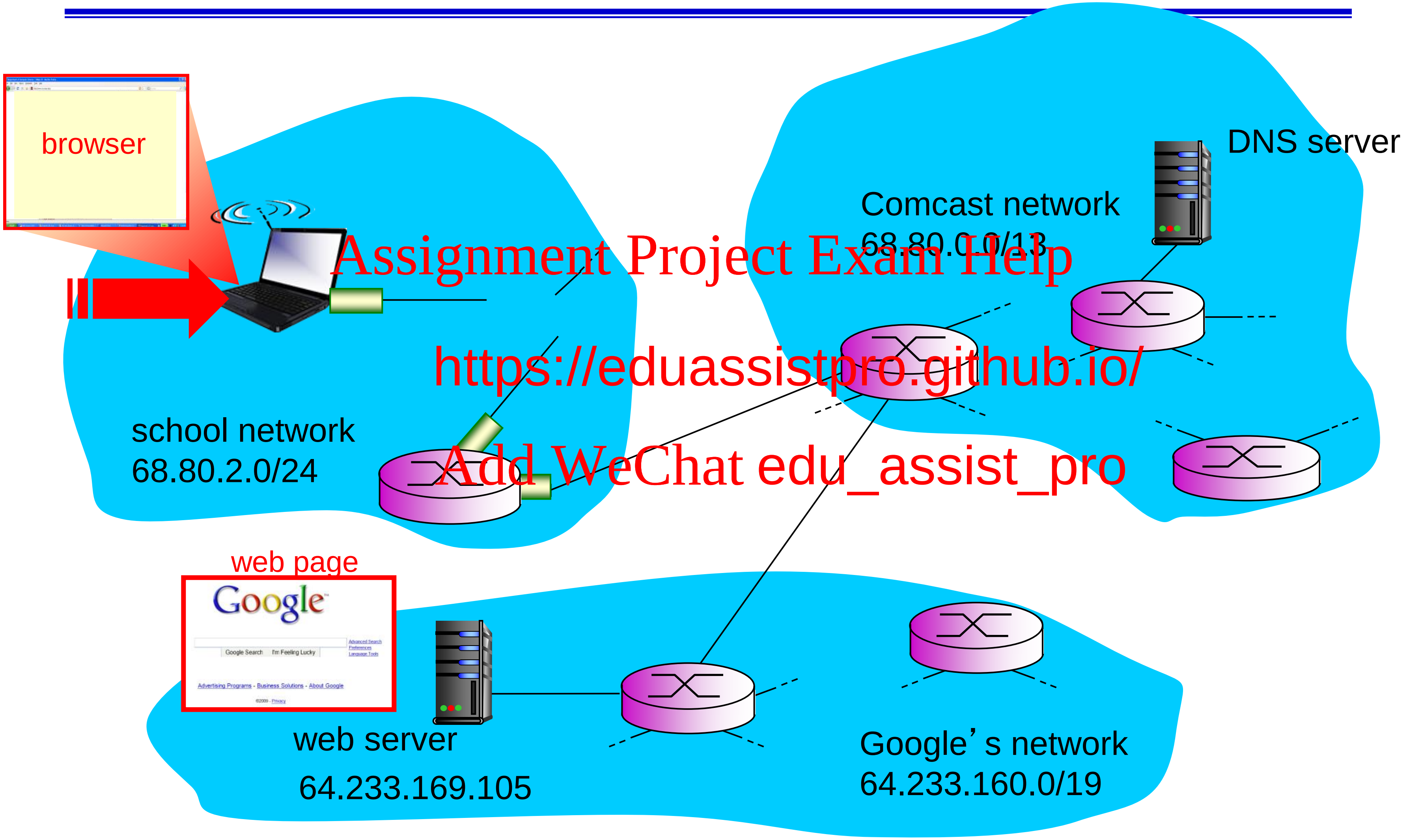
$$\text{Transmission efficiency} = \frac{\text{\# of information bits}}{\text{\# of information + overhead bits}}$$

FIGURE 4-12

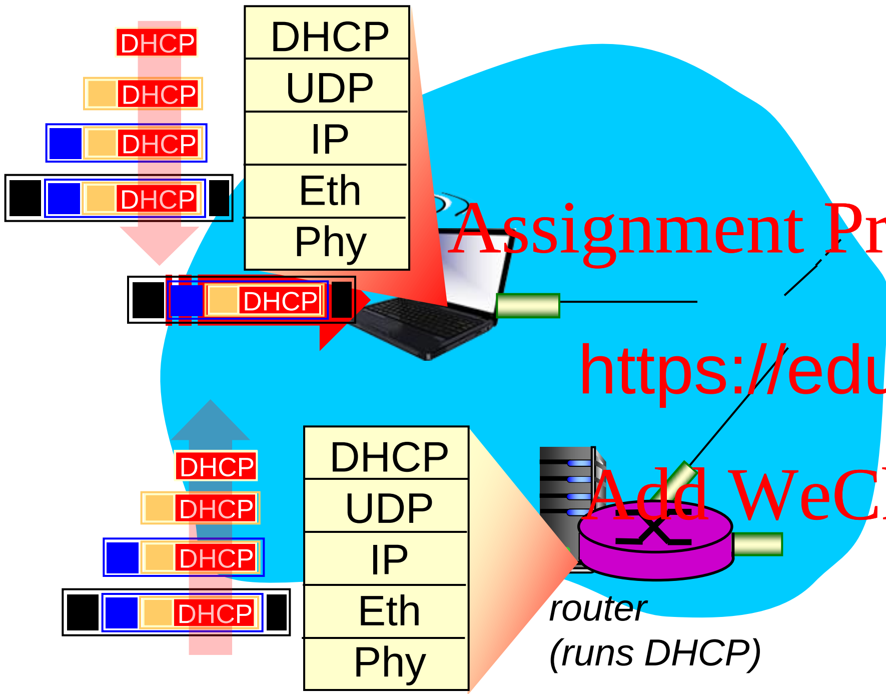
Frame size effects on throughput



A day in the life: scenario

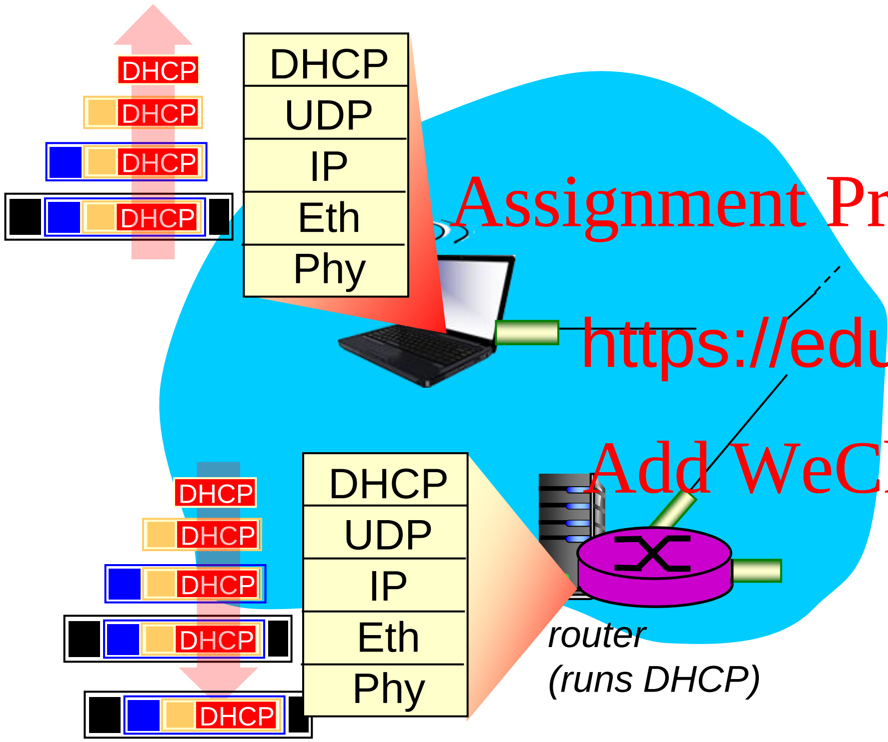


A day in the life… connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*
- DHCP request *encapsulated* in
  encapsulated in *IP*,
  sulated in *802.3* Ethernet
- t frame broadcast (dest: FFFFF) on LAN, received at router running DHCP server
- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

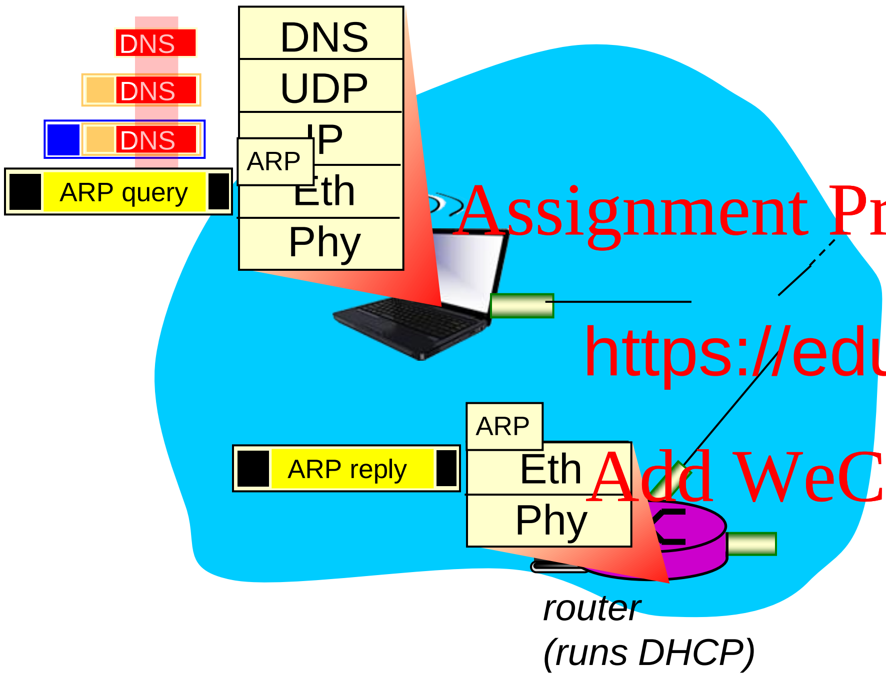
A day in the life… connecting to the Internet



- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- sulation at DHCP frame forwarded) through multiplexing at
- DHCP client receives DHCP ACK reply

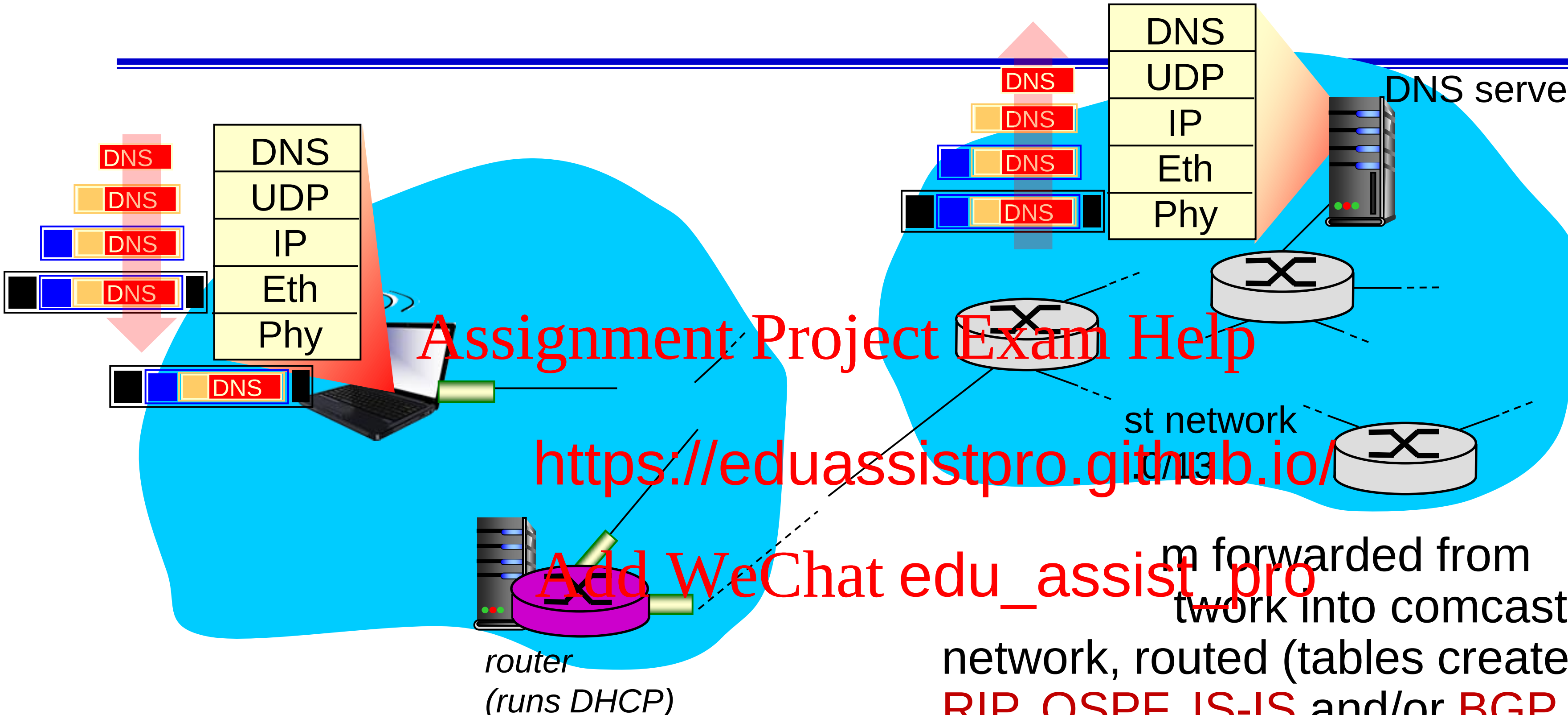
Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



- before sending HTTP request, need IP address of www.google.com: DNS
- DNS query created, encapsulated in UDP, encapsulated in IP, ed in Eth. To send frame need MAC address of rface: *ARP*
- roadcast, received by h replies with ARP reply giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life… using DNS

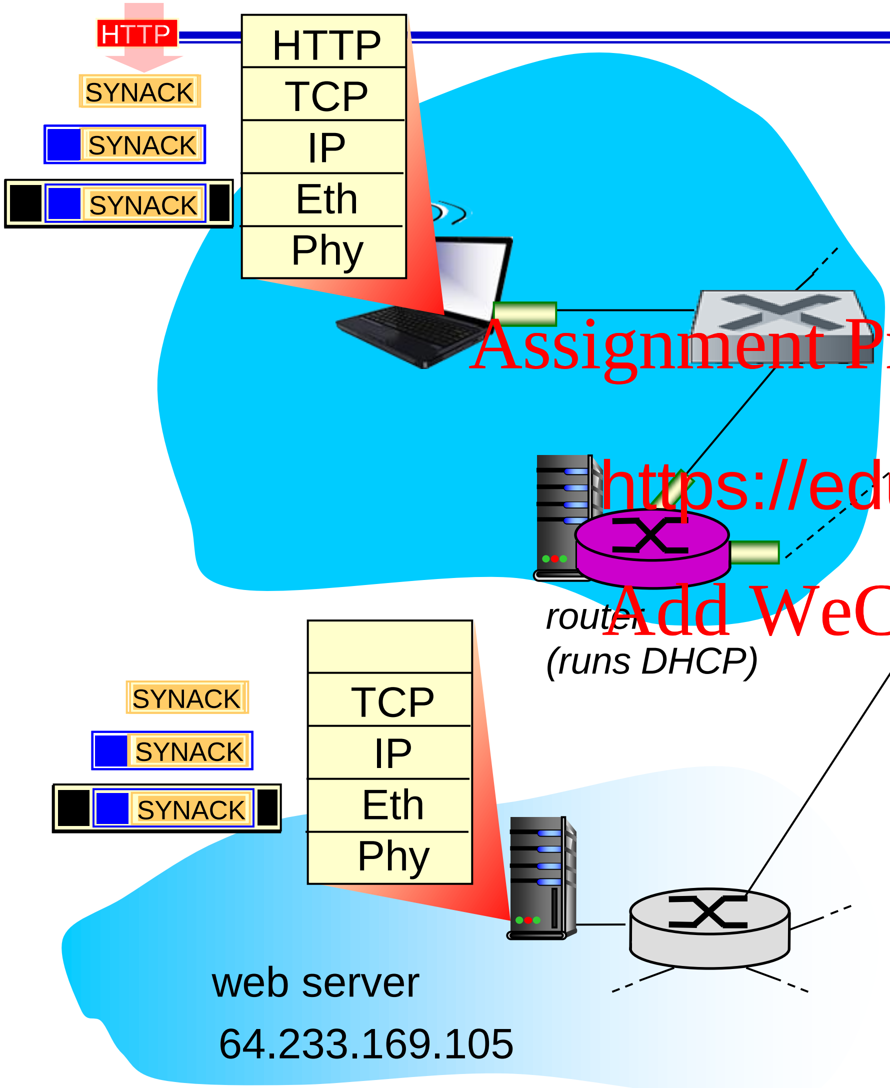


- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

m forwarded from
twork into comcast
network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server

- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

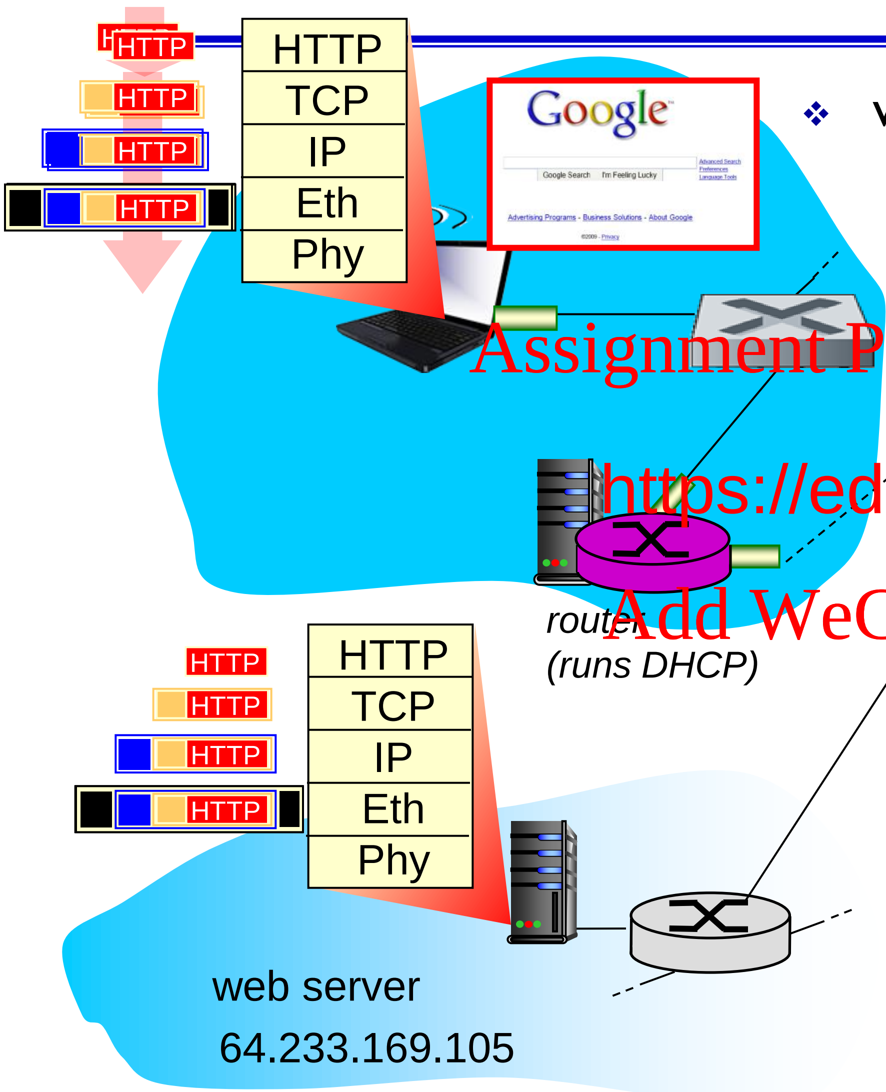
A day in the life...TCP connection carrying HTTP



nd HTTP request, nitiate *TCP handshake*

- TCP SYN segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in 3-way handshake)
- TCP connection established!

A day in the life... HTTP request/reply



- web page finally (!!!) displayed

P request sent into socket

gram containing HTTP t routed to www.google.com

- web server responds with HTTP reply (containing web page)
- IP datagram containing HTTP reply routed back to client

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro