



Network

Fundamentals of Ethernet LAN

So Ethernet?

Ethernet is everywhere

Packet data transmission = small blocks (Ethernet)

Addressing in every packet(frame)

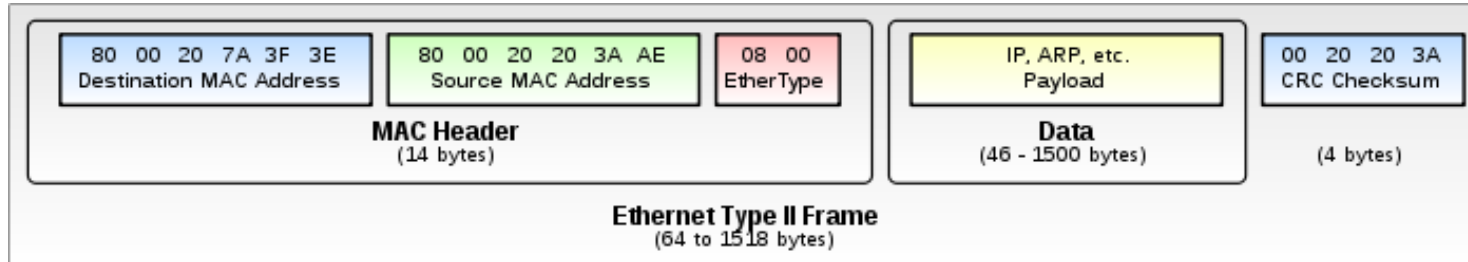
MTU(1500kbytes) in Ethernet

Addressing in Ethernet and IP

Ethernet		IP		DATA
dmac:00:0c:ce:13:b9:a0	smac:00:22:5f:98:91	sip:192.168.0.89	dip:192.168.0.1	



Ethernet frame

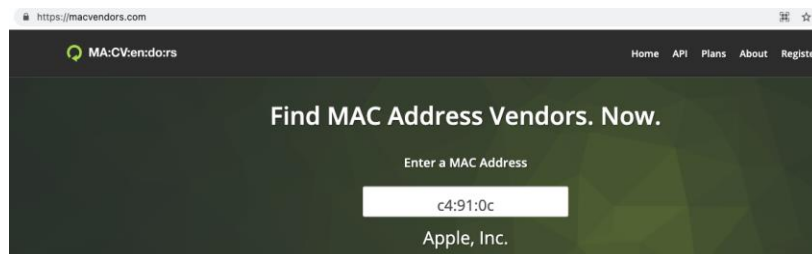


Next slide is about Ethertype

Payload = Data

Lets find a mac-address

80 00 20 20 3A AE	
Source MAC Address	
80 00 20 7A 3F 3E	
Destination MAC Address	
OUI	Vendor assigned



```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=400<CHANNEL_IO>
ether c4:91:0c:b2:e4:31
inet 192.168.1.65 netmask 0xffffffff broadcast 192.168.1.255
inet6 fe80::46f:541f:b95c:fb03%en0 prefixlen 64 secured scopeid 0xc
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
```

Ethernet frame

https://en.wikipedia.org/wiki/EtherType

Values [edit]

EtherType values for some notable protocols^[8]

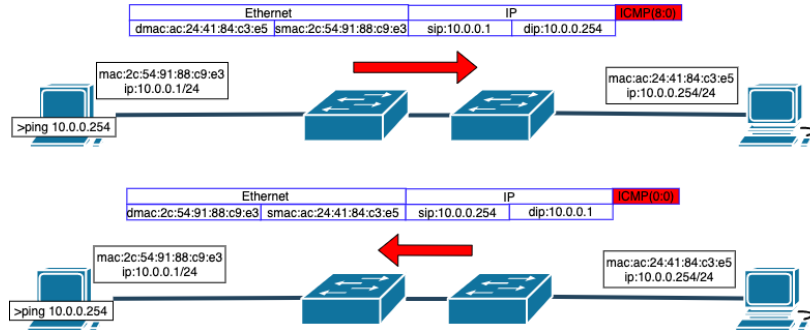
EtherType (hexadecimal)	Protocol
0x0800	Internet Protocol version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)
0x0842	Wake-on-LAN ^[9]
0x22F0	Audio Video Transport Protocol (AVTP)
0x22F3	IETF TRILL Protocol
0x22EA	Stream Reservation Protocol
0x6002	DEC MOP RC
0x6003	DECnet Phase IV, DNA Routing
0x6004	DEC LAT
0x8035	Reverse Address Resolution Protocol (RARP)
0x809B	AppleTalk (EtherTalk)
0x80F3	AppleTalk Address Resolution Protocol (AARP)
0x8100	VLAN-tagged frame (IEEE 802.1Q) and Shortest Path Bridging IEEE 802.1aq with NNI compatibility ^[10]
0x8102	Simple Loop Prevention Protocol (SLPP)
0x8103	Virtual Link Aggregation Control Protocol (VLACP)
0x8137	IPX
0x8204	QNX Qnet
0x86DD	Internet Protocol Version 6 (IPv6)
0x8808	Ethernet flow control
0x8809	Ethernet Slow Protocols ^[11] such as the Link Aggregation Control Protocol (LACP)
0x8819	CobraNet
0x8847	MPLS unicast
0x8848	MPLS multicast
0x8863	PPPoE Discovery Stage
0x8864	PPPoE Session Stage
0x887B	HomePlug 1.0 MME

https://en.wikipedia.org/wiki/EtherType

0x8819	CobraNet
0x8847	MPLS unicast
0x8848	MPLS multicast
0x8863	PPPoE Discovery Stage
0x8864	PPPoE Session Stage
0x887B	HomePlug 1.0 MME
0x888E	EAP over LAN (IEEE 802.1X)
0x8892	PROFINET Protocol
0x889A	HyperSCSI (SCSI over Ethernet)
0x88A2	ATA over Ethernet
0x88A4	EtherCAT Protocol
0x88A8	Service VLAN tag identifier (S-Tag) on Q-in-Q tunnel.
0x88AB	Ethernet Powerlink ^[citation needed]
0x88B8	GOOSE (Generic Object Oriented Substation event)
0x88B9	GSE (Generic Substation Events) Management Services
0x88BA	SV (Sampled Value Transmission)
0x88BF	MikroTik RoMON ^[9] (unofficial)
0x88CC	Link Layer Discovery Protocol (LLDP)
0x88CD	SERCOS III
0x88E3	Media Redundancy Protocol (IEC62439-2)
0x88E5	IEEE 802.1AE MAC security (MACsec)
0x88E7	Provider Backbone Bridges (PBB) (IEEE 802.1ah)
0x88F7	Precision Time Protocol (PTP) over IEEE 802.3 Ethernet
0x88F8	NC-SI
0x88FB	Parallel Redundancy Protocol (PRP)
0x8902	IEEE 802.1ag Connectivity Fault Management (CFM) Protocol / ITU-T Recommendation Y.1731 (OAM)
0x8906	Fibre Channel over Ethernet (FCoE)
0x8914	FCoE Initialization Protocol
0x8915	RDMA over Converged Ethernet (RoCE)
0x891D	TTEthernet Protocol Control Frame (TTE)
0x893a	1905.1 IEEE Protocol
0x892F	High-availability Seamless Redundancy (HSR)
0x9000	Ethernet Configuration Testing Protocol ^[12]
0xF1C1	Redundancy Tag (IEEE 802.1CB Frame Replication and Elimination for Reliability)

Ping adventure. Network is working

Devices are working for a long time



- 1) We can't send ICMP without IP header and Ethernet header
- 2) But how do we get Ethernet mac-addresses?

```
dns+icmp.pcapng
[Apply a display filter ...<36/>]
No. Time Source Destination Protocol Length Info
6 6.019290 192.168.43.9 8.8.8.8 ICMP 98 Echo (ping) request id=0xd73b, seq=1/256, ttl=64 (reply in 7)

> Frame 6: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en1, id 0
> Ethernet II, Src: Apple_13:c5:58 (60:33:4b:13:c5:58), Dst: MS-NLB-PhysServer-26_11:f0:c8:3b (02:1a:11:f0:c8:3b)
> Internet Protocol Version 4, Src: 192.168.43.9, Dst: 8.8.8.8
> Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xb768 [correct]
  [Checksum Status: Good]
  Identifier (BE): 55099 (0xd73b)
  Identifier (LE): 15319 (0x3bd7)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
  [Response frame: 7]
  Timestamp from icmp data: May 31, 2013 02:45:18.284205000 MSK
  [Timestamp from icmp data (relative): 0.004938000 seconds]
> Data (48 bytes)
```

```
dns+icmp.pcapng
[Apply a display filter ...<36/>]
No. Time Source Destination Protocol Length Info
6 6.019290 192.168.43.9 8.8.8.8 ICMP 98 Echo (ping) request id=0xd73b, seq=1/256, ttl=64 (reply in 7)
7 6.153053 8.8.8.8 192.168.43.9 ICMP 98 Echo (ping) reply id=0xd73b, seq=1/256, ttl=64 (request in 6)

> Frame 7: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en1, id 0
> Ethernet II, Src: MS-NLB-PhysServer-26_11:f0:c8:3b (02:1a:11:f0:c8:3b), Dst: Apple_13:c5:58 (60:33:4b:13:c5:58)
> Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.43.9
> Internet Control Message Protocol
  Type: 0 (Echo (ping) reply)
  Code: 0
  Checksum: 0xb768 [correct]
  [Checksum Status: Good]
  Identifier (BE): 55099 (0xd73b)
  Identifier (LE): 15319 (0x3bd7)
  Sequence Number (BE): 1 (0x0001)
  Sequence Number (LE): 256 (0x0100)
  [Request frame: 6]
  [Response time: 134.363 ms]
  Timestamp from icmp data: May 31, 2013 02:45:18.284205000 MSK
  [Timestamp from icmp data (relative): 0.139301000 seconds]
> Data (48 bytes)
```

ARP

ARP its a protocol for getting DMAC from DIP

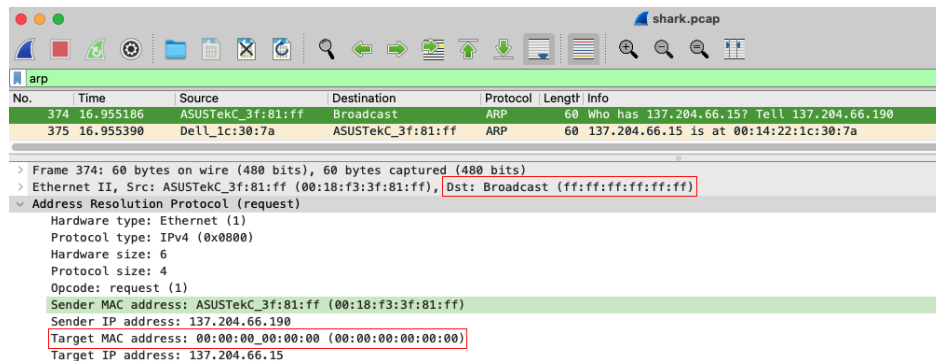
Arp -a (windows, linux command)

Every router, PC/server, or any what has got an IP address should fill ethernet headers to send info via NIC. So its should have ARP table

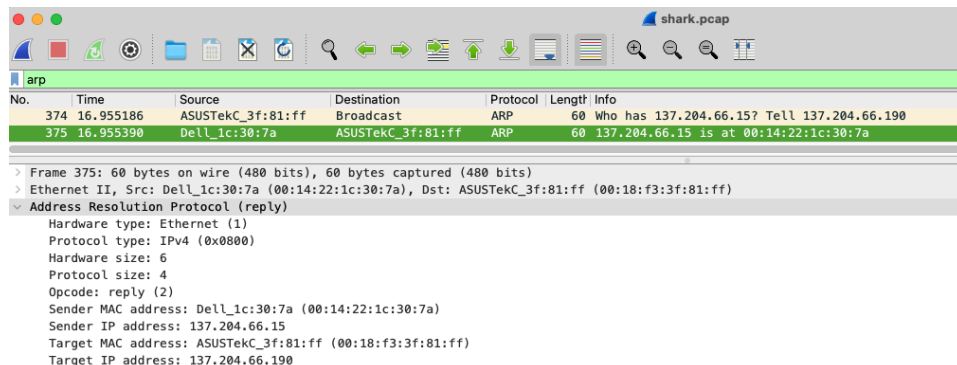
ARP entries could be static, dynamic, dynamic entries have timers (default 4hours)

223.1.2.1	08:00:39:00:2F:C3
223.1.2.3	08:00:5A:21:A7:22
223.1.2.4	08:00:10:99:AC:54

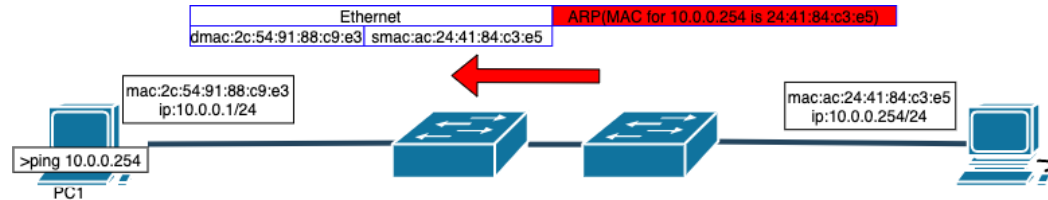
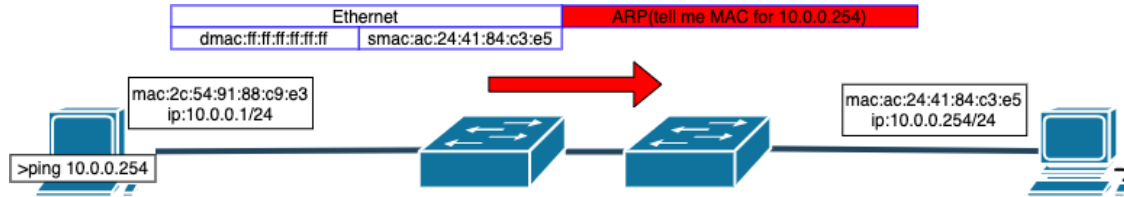
ARP wireshark



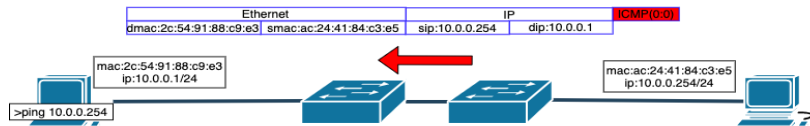
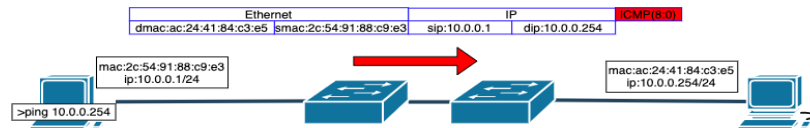
ARPs don't have an IP header
They can't be routed.
They can't be via a Router.



Ping adventure 2. PC were rebooted



PC1:
Arp -a
10.0.0.254 24:41:84:c3:e5



From 'ping adventure 1' slide

Ethernet could be different

Speed/Standard

10mb

100mb

1gb

10gb

40gb

100gb

100meters?

Copper/Fiber

Half-duplex vs Full-duplex

Full-duplex faster than half-duplex

In Full-duplex we don't have collisions (in Half-duplex we have collisions)

Collisions in half duplex



Hi, how are you doing?

(same time)

Hi, nice to see you!

What did you say?

Sorry, what?



Half-duplex vs Full-duplex

Full-duplex

Hi, how are you doing? (same time) Hi, nice to see you!

Hey, hey, I'm doing well. I'm okay.



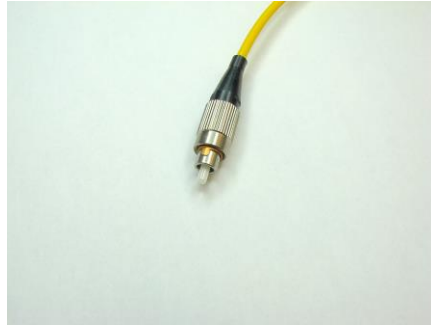
Mouth

Ear

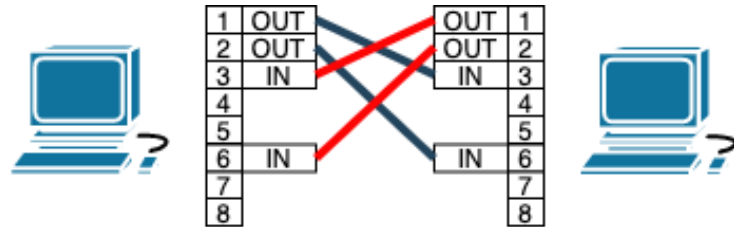


Cables

Crossover vs Straight-through



Cables

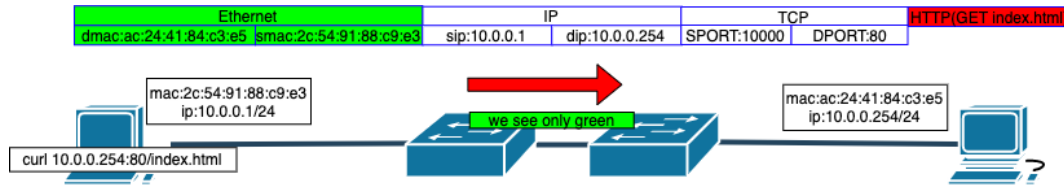
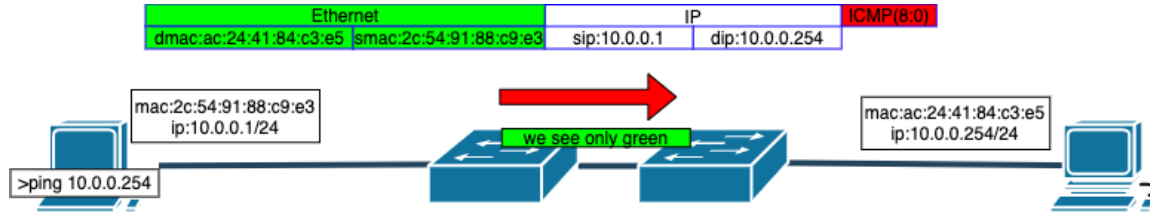


Crossover (for pc-pc, with hub)



Straight-through (with switches, routers)

Point of view



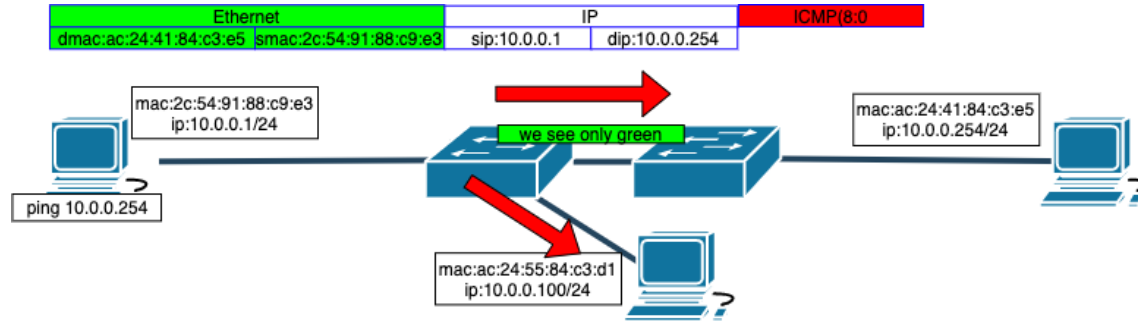
Switches are not smart?

Well, switches, hubs, bridges see only ethernet headers(mac addressing, ethertype)

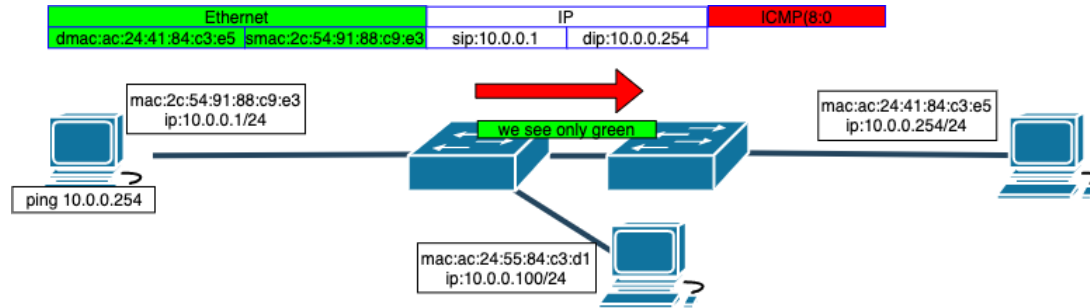
They are only forwarding Ethernet, they see ONLY ethernet.

Hub, Switches, Bridges

Hubs send to every interface except that interface, a packet has come

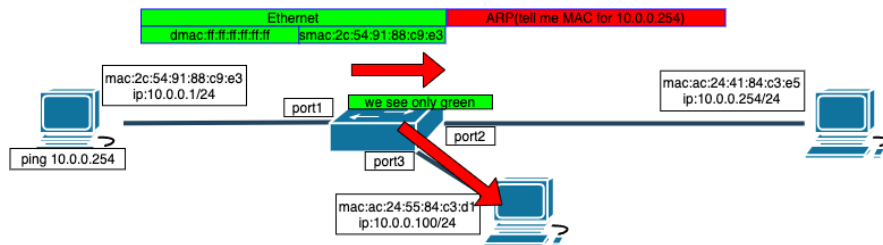


Switches send to interface except that interface. But how do they know? CAM/MAC table

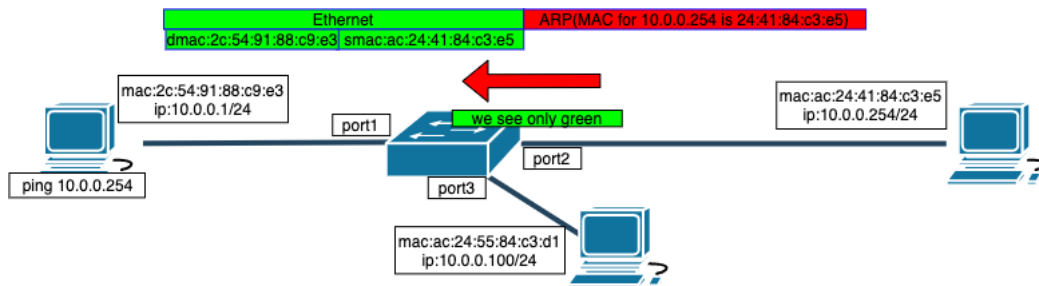


MAC table

ARP



Switch MAC table:
2c:54:91:88:c9:e3 on port1 (new entry)



Switch MAC table:
2c:54:91:88:c9:e3 on port1
ac:24:41:84:c3:e5 on port2 (new entry)

Switches learn SMAC and add them to the MAC table.
Why? To be the switches, not hubs :-)

MAC table



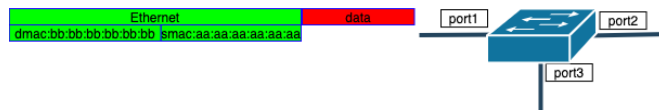
MAC table:
no entries



MAC table:
no entries



MAC table:
aa:aa:aa:aa:aa:aa port 2



MAC table:
cd:cd:cd:cd:cd:cd port 2
bb:bb:bb:bb:bb:bb port 2
ee:ee:ee:ee:ee:ee port 2
ab:ab:ab:ab:ab:ab port 1
aa:aa:aa:aa:aa:aa port 1

Hub or Switch?

If thing has got MAC table – its a Switch

If thing doesn't have MAC table – its a Hub

Wireshark Ethernet

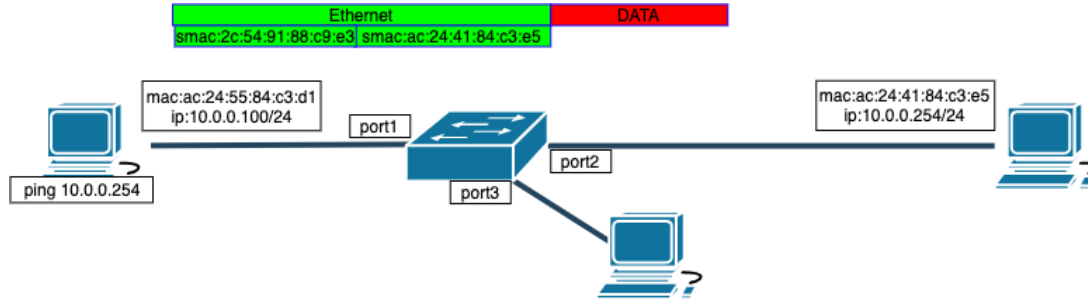
The image shows a Wireshark packet capture window titled "http.cap". The packet list on the left shows 22 packets. Packet 17 is selected, showing details of an Ethernet II frame, an Internet Protocol Version 4 packet, a User Datagram Protocol packet, and a Domain Name System (response) packet. The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=5521 Win=9660 Len=0
13	2.553672	145.254.160.237	145.253.2.203	DNS	89	Standard query 0x0023 A pagead2.googlesyndication.com
14	2.633787	65.208.228.223	145.254.160.237	TCP	1434	80 → 3372 [ACK] Seq=5521 Ack=480 Win=6432 Len=1380 [TCP segment of a reassembled PDU]
15	2.814046	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=6901 Win=9660 Len=0
16	2.894161	65.208.228.223	145.254.160.237	TCP	1434	80 → 3372 [ACK] Seq=6901 Ack=480 Win=6432 Len=1380 [TCP segment of a reassembled PDU]
17	2.914190	145.253.2.203	145.254.160.237	DNS	188	Standard query response 0x0023 A pagead2.googlesyndication.com CNAME pagead2.google.com CNAME pagead.g...
18	2.984291	145.254.160.237	216.239.59.99	HTTP	775	GET /pagead/ads?client=ca-pub-2309191948673629&random=1084443430285&mt=1082467020&format=468x60_as&ou...
19	3.014334	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=8281 Win=9660 Len=0
20	3.374852	65.208.228.223	145.254.160.237	TCP	1434	80 → 3372 [ACK] Seq=8281 Ack=480 Win=6432 Len=1380 [TCP segment of a reassembled PDU]
21	3.495025	65.208.228.223	145.254.160.237	TCP	1434	80 → 3372 [PSH, ACK] Seq=9661 Ack=480 Win=6432 Len=1380 [TCP segment of a reassembled PDU]
22	3.495025	145.254.160.237	65.208.228.223	TCP	54	3372 → 80 [ACK] Seq=480 Ack=11041 Win=9660 Len=0

> Frame 17: 188 bytes on wire (1504 bits), 188 bytes captured (1504 bits)
▼ Ethernet II, Src: fe:ff:20:00:01:00 (fe:ff:20:00:01:00), Dst: Xerox_00:00:00 (00:00:01:00:00:00)
 > Destination: Xerox_00:00:00 (00:00:01:00:00:00)
 > Source: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
 Type: IPv4 (0x0800)
 > Internet Protocol Version 4, Src: 145.253.2.203, Dst: 145.254.160.237
 > User Datagram Protocol, Src Port: 53, Dst Port: 3009
 > Domain Name System (response)

```
0000  00 00 01 00 00 00 fe ff 20 00 01 00 08 00 45 00  .....E.....
0010  00 ae 15 95 40 00 f9 11 a3 f5 91 fd 02 cb 91 fe  ....@.....
0020  a0 ed 00 35 0b c1 00 9a 30 02 00 23 81 80 00 01  ...5....0.#.....
0030  00 04 00 00 00 00 07 70 61 67 65 61 64 32 11 67  ....p agead2 g
0040  6f 6f 67 6c 65 73 79 6e 64 69 63 61 74 69 6f 6e  ...oglesyn dication
0050  03 63 6f 6d 00 00 01 00 01 c0 0c 00 05 00 01 00  .com.....
0060  00 bc c1 00 11 07 70 61 67 65 61 64 32 06 67 6f  ....pa gead2 go
0070  6f 67 6c 65 c0 26 c0 3b 00 05 00 01 00 00 7a  ...ogle & ; .....z
0080  00 1a 06 70 61 67 65 61 64 06 67 6f 67 6c 65  ...pagea d-google
0090  06 61 6b 61 64 6e 73 03 6e 65 74 00 c0 58 00 01  ...akadns net; X..
00a0  00 01 00 00 00 7b 00 04 d8 ef 3b 68 c0 58 00 01  ....{...;h X..
00b0  00 01 00 00 7b 00 04 d8 ef 3b 63  ....{...;c
```

Lets do it again?



1) We have Ethernet NIC

Any ICMP/HTTP/HTTPS/TCP/UDP will be encapsulated to IP and to Ethernet.

2)PC should fill the ETHERNET headers. PC should know DMAC(uses ARP)

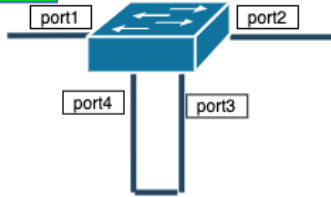
3)Switch uses MAC table to send ETHERNET frames

4)PC has got 1 table: ARP table (matching ip to DMAC). ARP table is showing WHERE

5)Switch has got 1 table: MAC table (matching SMAC to interface). MAC table is showing WHERE and WHERE FROM

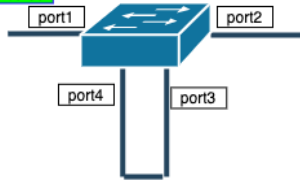
Ethernet Loop?

Ethernet
dmac:aa:aa:aa:aa:aa:aa smac:ac:24:41:84:c3:e5



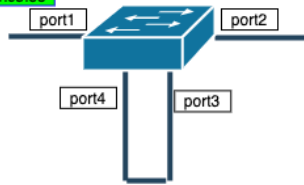
MAC table:
aa:aa:aa:aa:aa:aa port 2

Ethernet
dmac:bb:bb:bb:bb:bb:bb smac:ac:24:41:84:c3:e5



MAC table:
aa:aa:aa:aa:aa:aa port 2

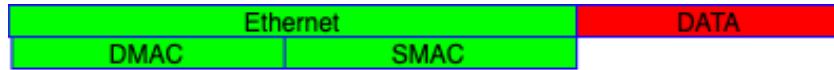
Ethernet
dmac:ff:ff:ff:ff:ff:ff smac:ac:24:41:84:c3:e5



MAC table:
aa:aa:aa:aa:aa:aa port 2

FF:FF:FF:FF:FF:FF for port possible?
NO!

Ethernet addresses types



BROADCAST – where DMAC is FF:FF:FF:FF:FF:FF

MULTICAST – where DMAC is 01:00:5e:xx:xx:x

UNICAST – where DMAC is something else (aa:aa:aa:aa:aa:aa, 1b:2c:3c:11:[db:fe](#) etc)

UNKNOWN UNICAST – switch doesn't got THAT unicast DMAC – so send to all ports (except the received port) – like BROADCAST

How to get Ethernet address

```
[stepan@sun /]$ ifconfig
enp0s13f0u1: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 94:05:bb:14:45:7e txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
[stepan@sun /]$ ifconfig
enp0s13f0u1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.78 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::71b7:eae0:bd4e:d798 prefixlen 64 scopeid 0x20<link>
    ether 94:05:bb:14:45:7e txqueuelen 1000 (Ethernet)
    RX packets 5345 bytes 7064496 (6.7 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3946 bytes 342580 (334.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```


THANK YOU