SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Lec9 – The Link Layer

50.012 Networks

Jit Biswas

Cohort 1: TT7&8 (1.409-10)

Cohort 2: TT24&25 (2.503-4)

Introduction

- Todays lecture:
  - Overview link layer
  - Ethernet header, MAC addresses, and ARP
  - Switches revisited, VLANs
  - Medium Access Control
- Note: This slide set is based on Kurose & Ross chapter 5 slide set

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

The Link Layer

- Provides single-hop connection for *frames* between interfaces
  - Protocols: 802.3 Ethernet, 802.11 Wlan, ARP
  - Addressing is based on MAC addresses
- Remember:
  - App layer: connection between applications
  - Transport layer: connection between processes
  - Network layer: connections between hosts
- Main services of Link layer:
  - Addressing, framing, medium access control (MAC)
  - Error detection & correction

Link layer services

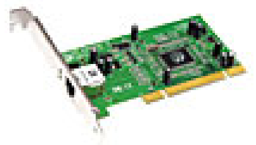
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

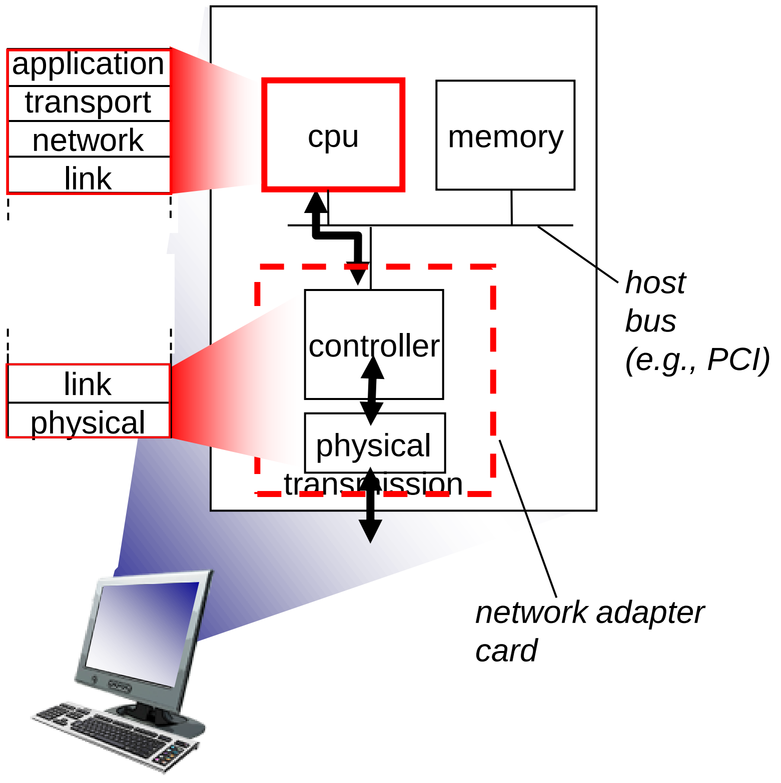
Link layer services (more)

- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

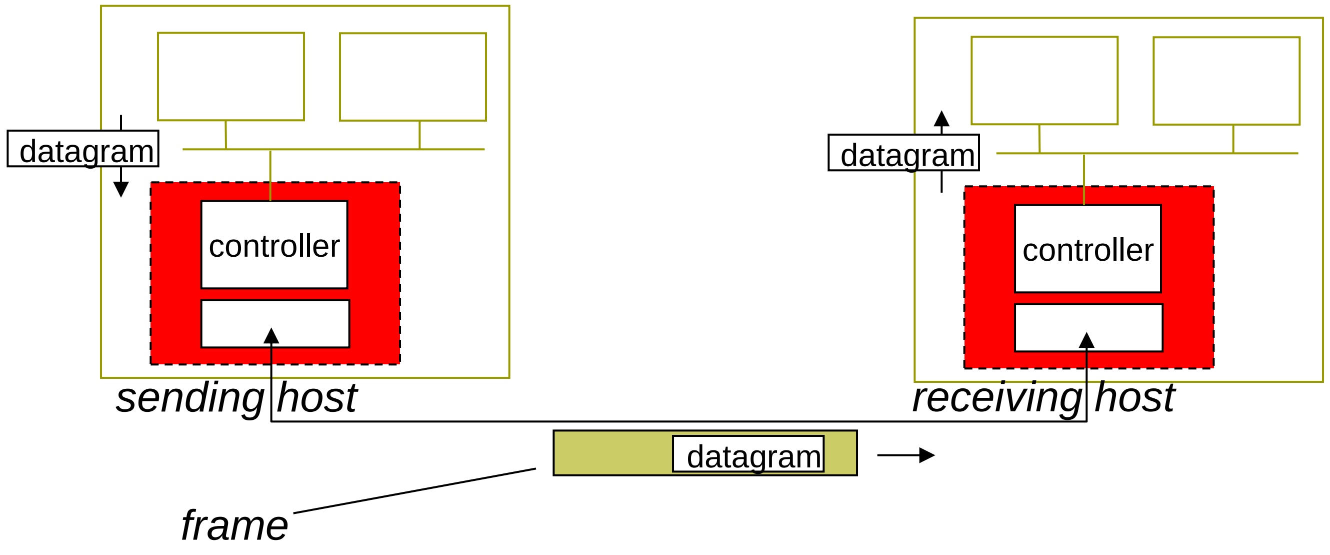
Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware





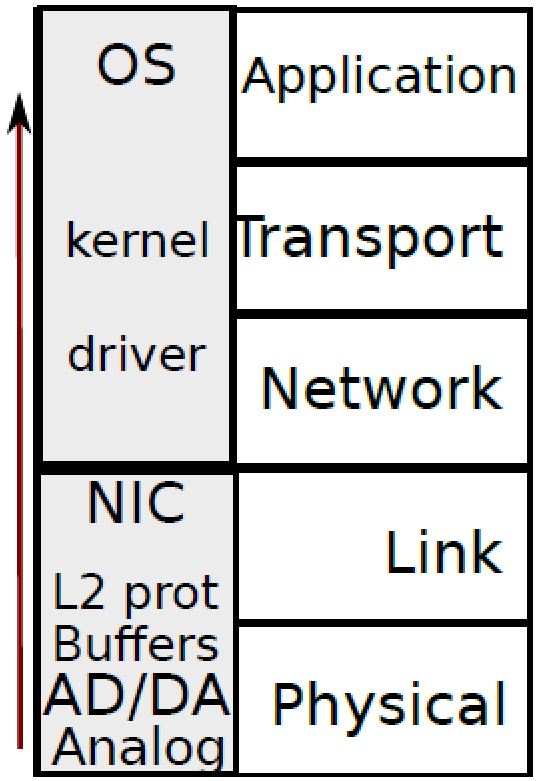
Adaptors communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Network Interfaces

- Netw. interface card (NIC) handles L1+L2
  - Application layer data is provided to the operating system/ processes
- NIC is processing in soft-/ or hardware
- Memory for input buffers
  - (e.g., IP and TCP fragments)
- Memory for output buffers
  - (e.g. TCP non-acked fragments)



IEEE 802.1, 802.3, 802.3, …

- Numbers denote working groups within IEEE
  - subversion numbers usually denote standards
- 802.1 is working on the following (and more)
  - 802 architectures, security
  - Internetworking between different 802 networks
- 802.2 is working on Link Layer Control (LLC)
  - Essentially, different Link Layer Headers
- 802.3 is working on wired LANs/ Ethernet
  - 802.3u is normal 100Mbit ethernet
- 802.11 is wireless working group
  - 802.11 (a,b,g,n) are different wireless standards

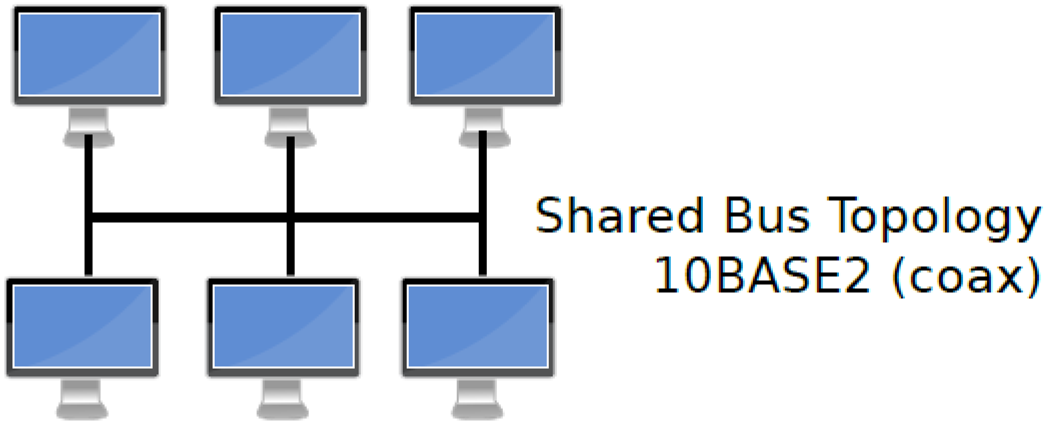
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

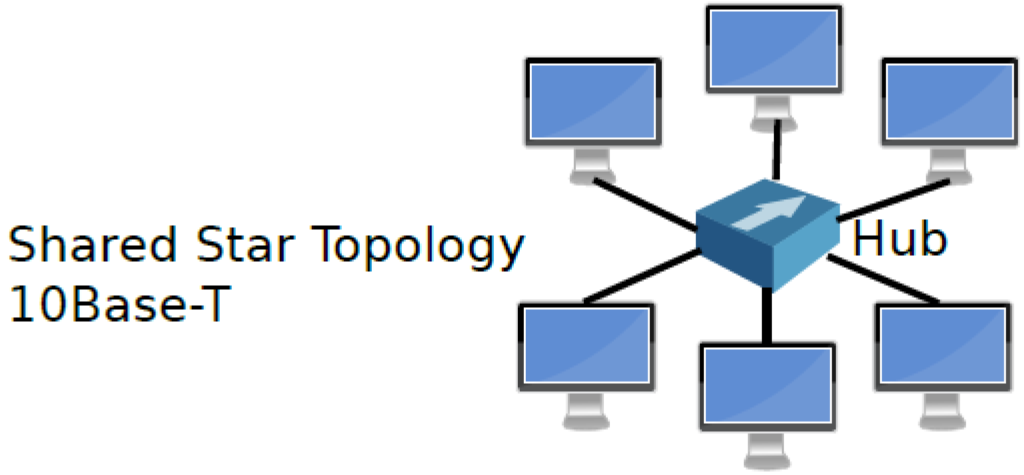
Ethernet

IEEE 802.3: Wired Ethernet

- The de-facto standard for wired LANs
- 802.3 specifies properties of Link and PHY layer
- Many physical layer variants (fibre, copper)
- Most common use: 802.3u, cheap twisted pair copper wires
- Originally started with 10Mbit/s
  - Coaxial cables, bus/ring topology
  - Shared medium, hubs (L1 repeaters)
- Extended over time, 10Gbit/s and more now
- Nowadays: Star architecture with L2+ switch
  - Medium not shared anymore, all peer-to-peer

Ethernet Shared Medium Topologies

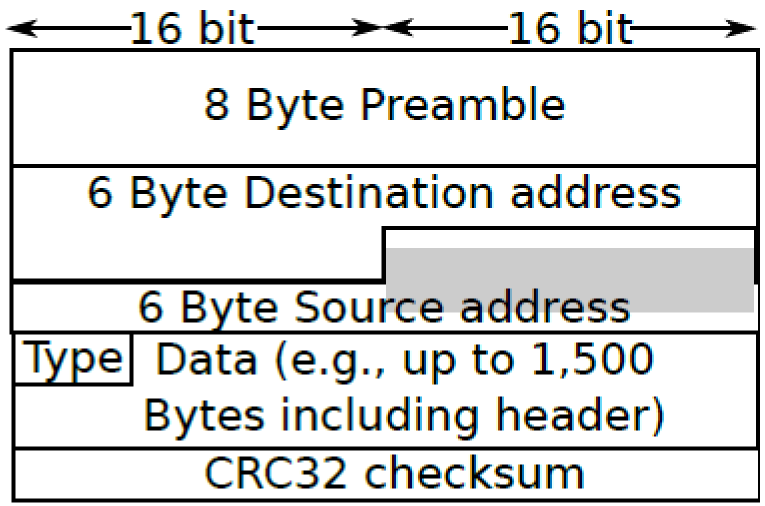




Ethernet Protocol

- Ethernet consists of different 802.3X standards
  - Copper/Fibre for PHY, headers very similar
- Connectionless - Like UDP, no handshakes etc.
- Unreliable - No retransmission protocol, but CRC
  - Cyclic Reduncancy Checksum allows to detect errors
  - Algorithm is based on math (GF(), see security lecture)
- Medium Access - Wired Ethernet can handle collisions
  - Without full duplex: CSMA/CD with exponential backoff
  - More on that later (PHY Layer)
- Why is a shared medium a problem/ how can it be solved?

Ethernet Frame overview


16 bit
16 bit
8 Byte Preamble
6 Byte Destination address
6 Byte Source address
Type
Data (e.g., up to 1,500 Bytes including header)
CRC32 checksum

- 802.3 (Ethernet II) header provides PHY-specific services
- *Preamble* required for sender/receiver synchronization
- Error detection: *cyclic redundancy checks*, e.g., CRC32
- Optionally: Flow control, error correction, etc (e.g., WLAN)
- Preamble and CRC are not provided to wireshark by interface

Ethernet Addresses

- Also called MAC addresses, identify *interface adapter*
  - 48 bit (IPv4:32 bit)
  - Example: 5e:3e:c1:32:d1:41
- Supposed to be globally unique
  - Must be unique in local network
  - Assigned by manufacturer
  - Manufacturers get sub-space assigned
- Addresses meant to be permanent (lifetime)
  - In some OS you can change MAC if you want

Address Resolution Protocol

MAC addresses and ARP

- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - function: *used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

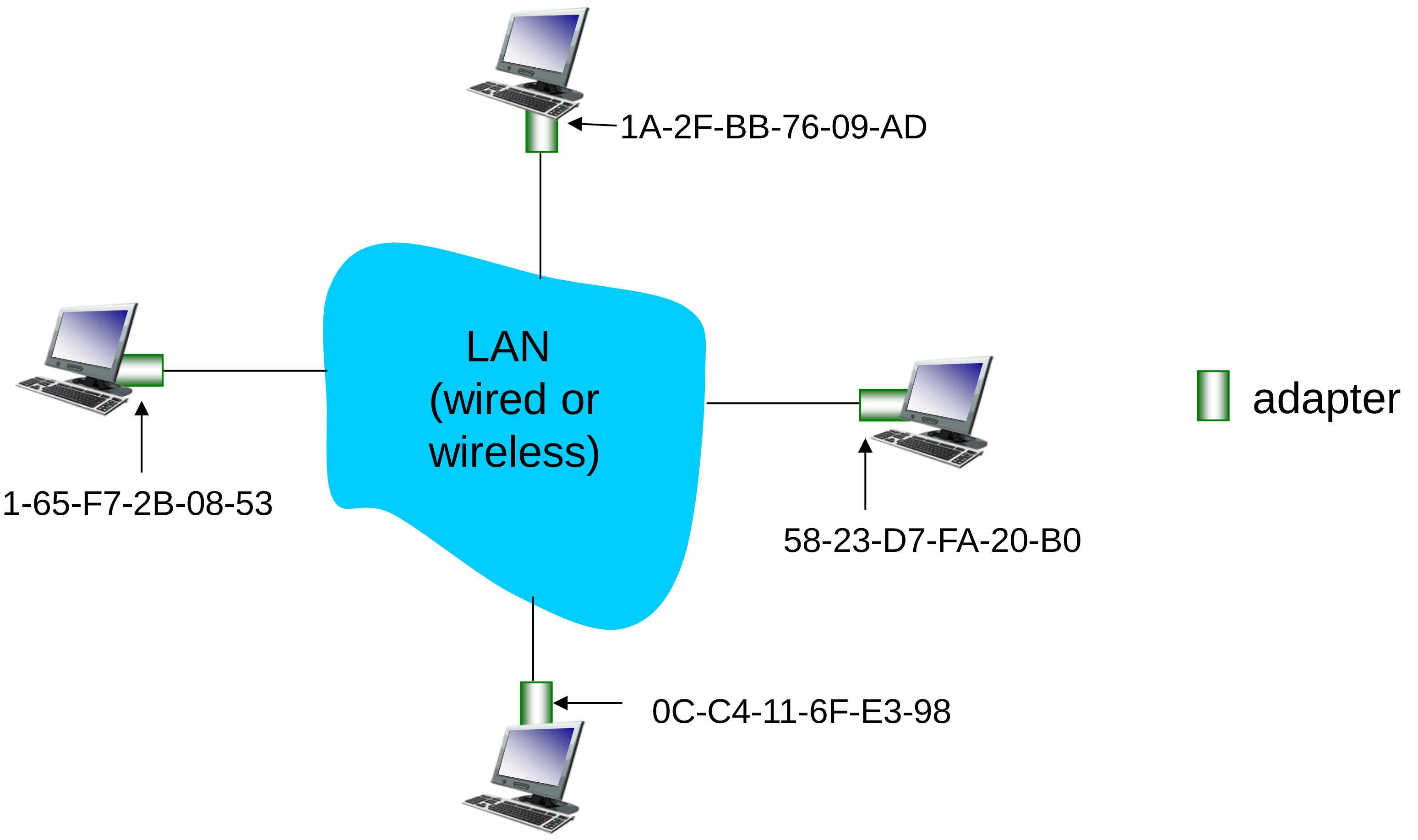
hexadecimal (base 16) notation
(each "number" represents 4 bits)

Address Resolution Protocol (ARP)

- How are URLs/IP+Port/IP mapped to MAC address?
- On network layer, next hop on path is found
  - This yields and IP address
- The sender then needs to find the MAC address of that host
  - Done with a simple broadcast request using ARP protocol
  - Target IP host will answer with his MAC
- MAC-addresses/ARP are only used in local networks
  - Single-hop, e.g., with Layer 2 switches
- Try for yourself (on Linux): arp will show your cache
  - Or the newer ip -s neighbor show

LAN addresses and ARP

each adapter on LAN has unique *LAN* address

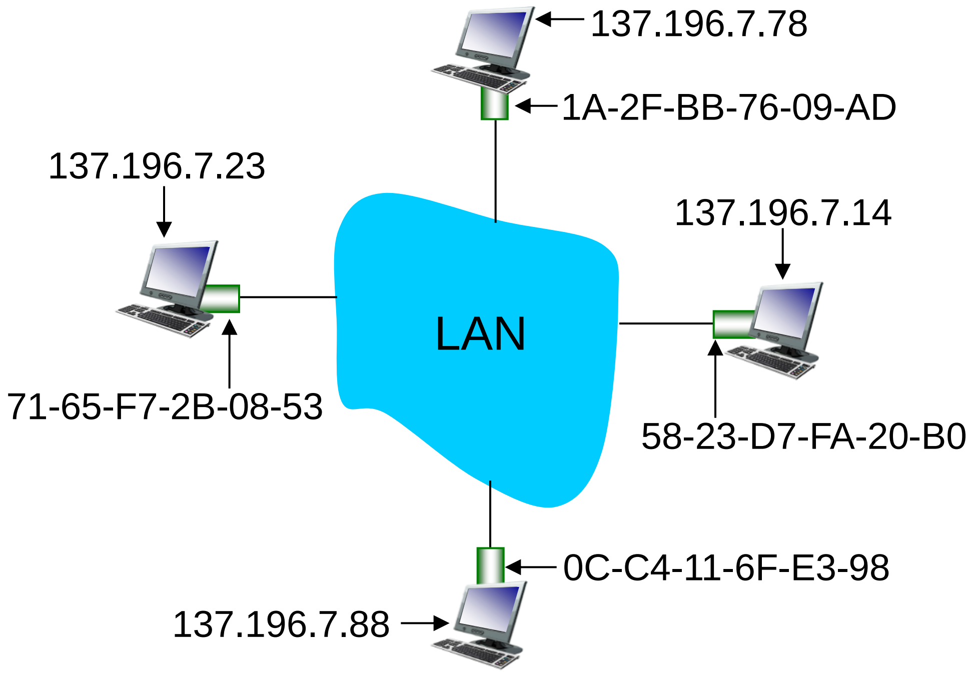


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached

ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

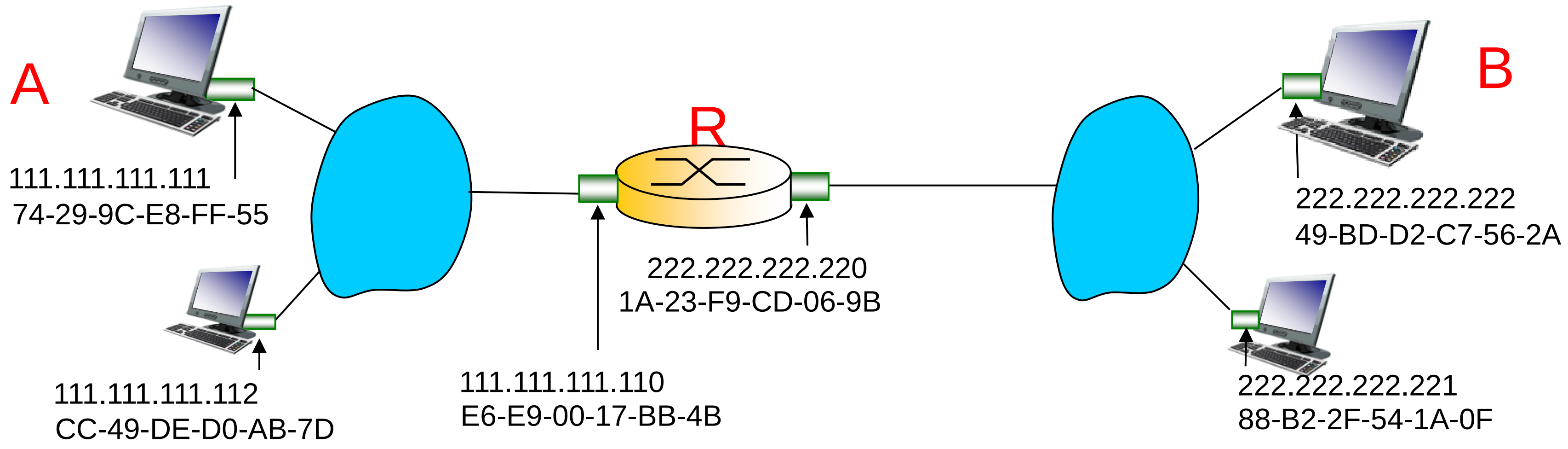
ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

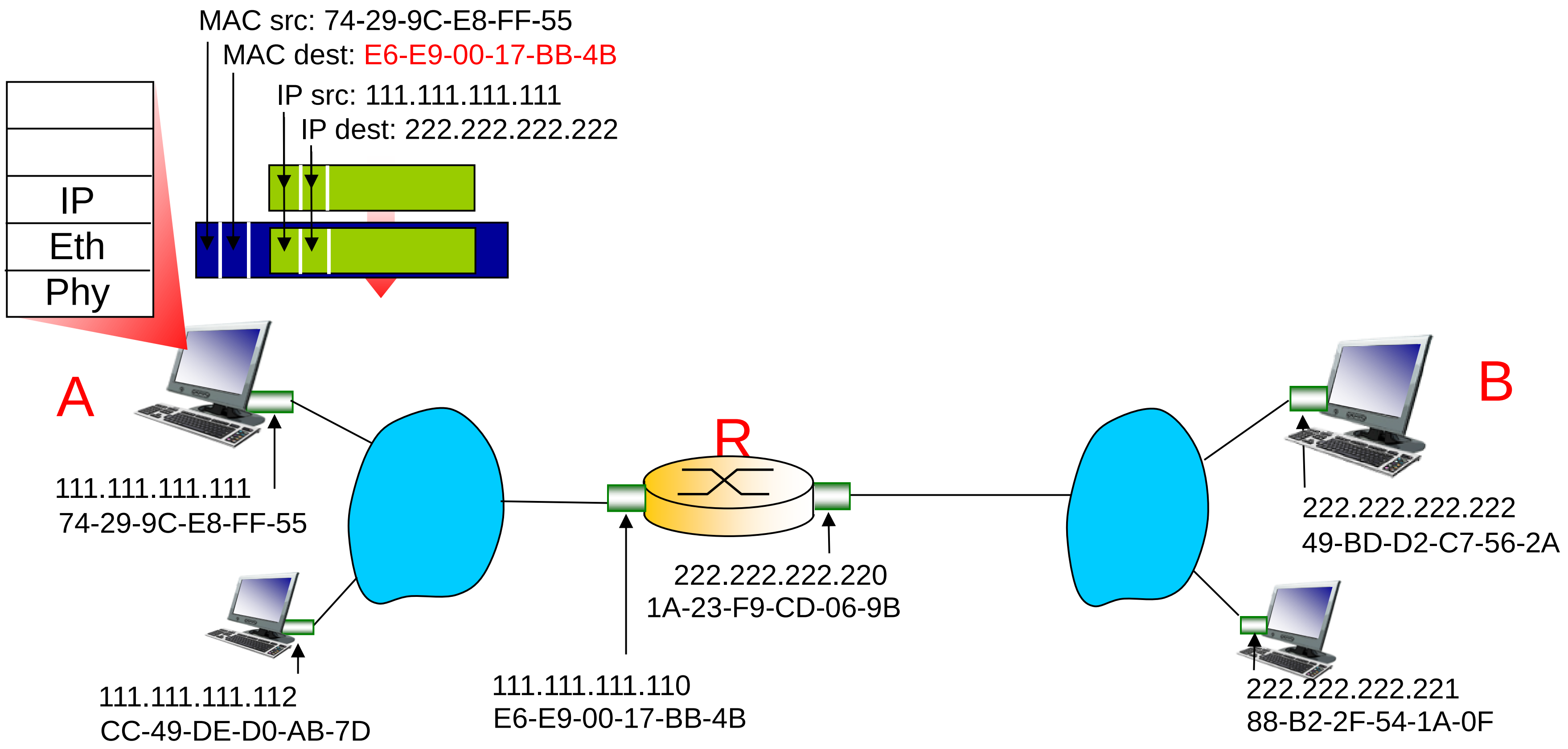
walkthrough: send datagram from A to B via R

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



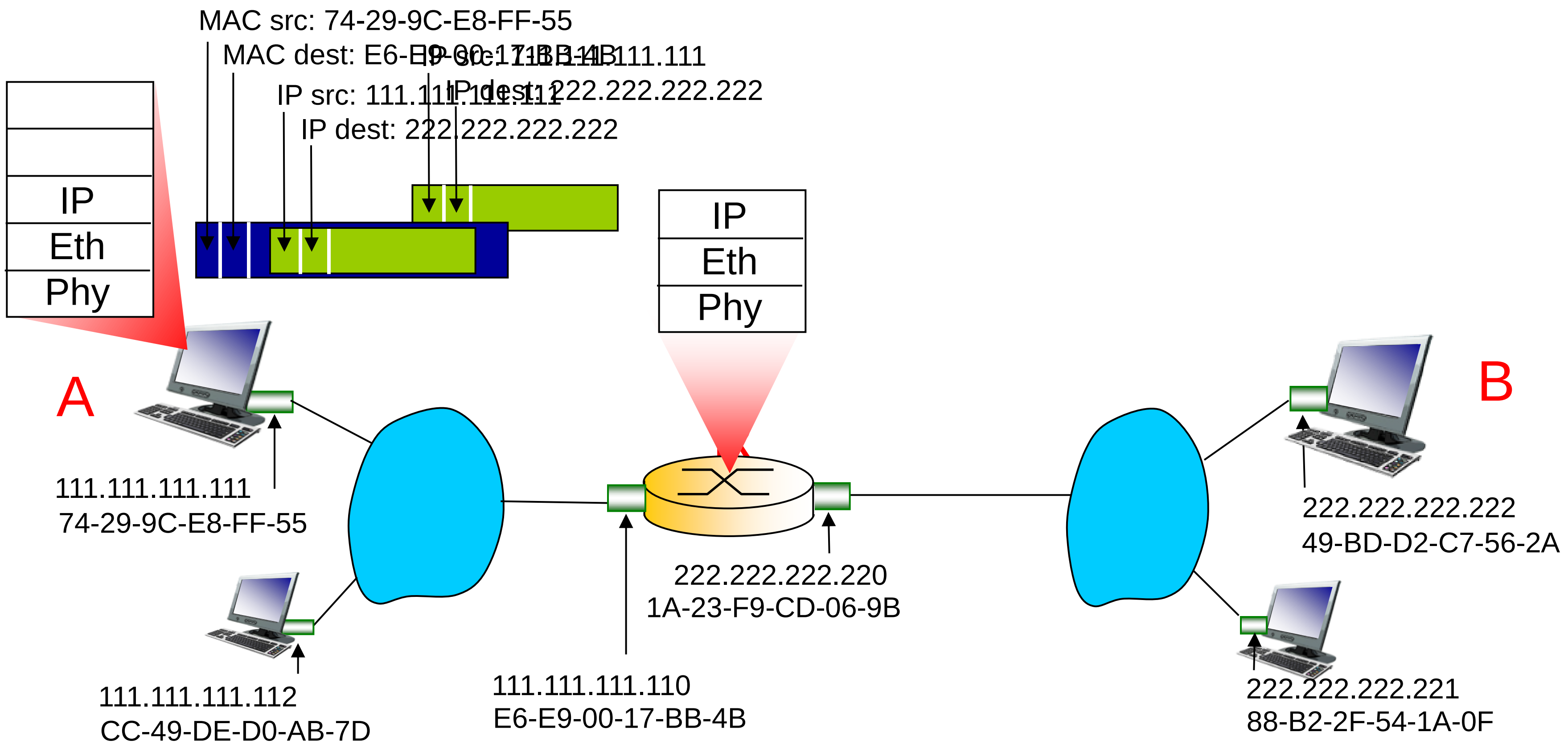
Addressing: routing to another LAN

- 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



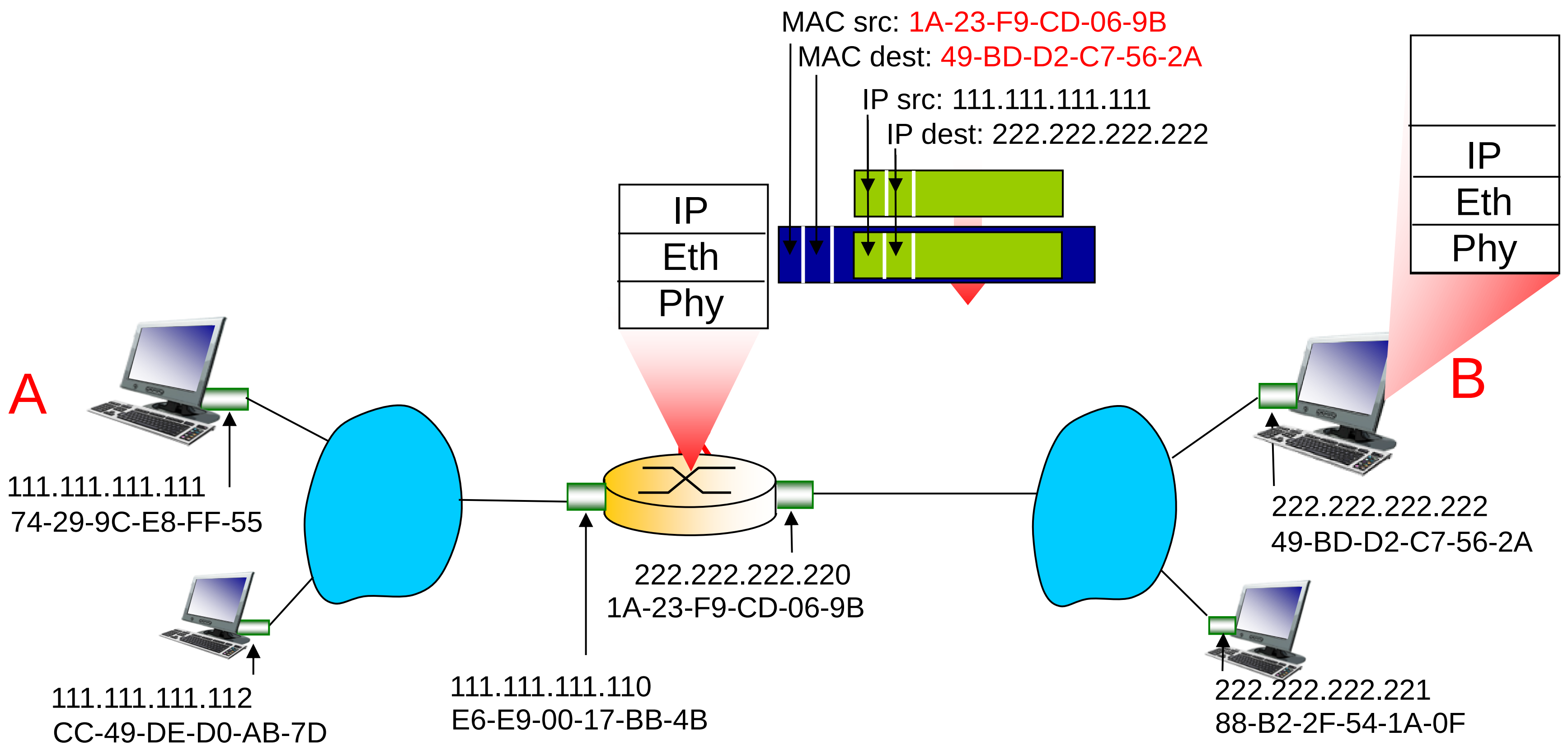
Addressing: routing to another LAN

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



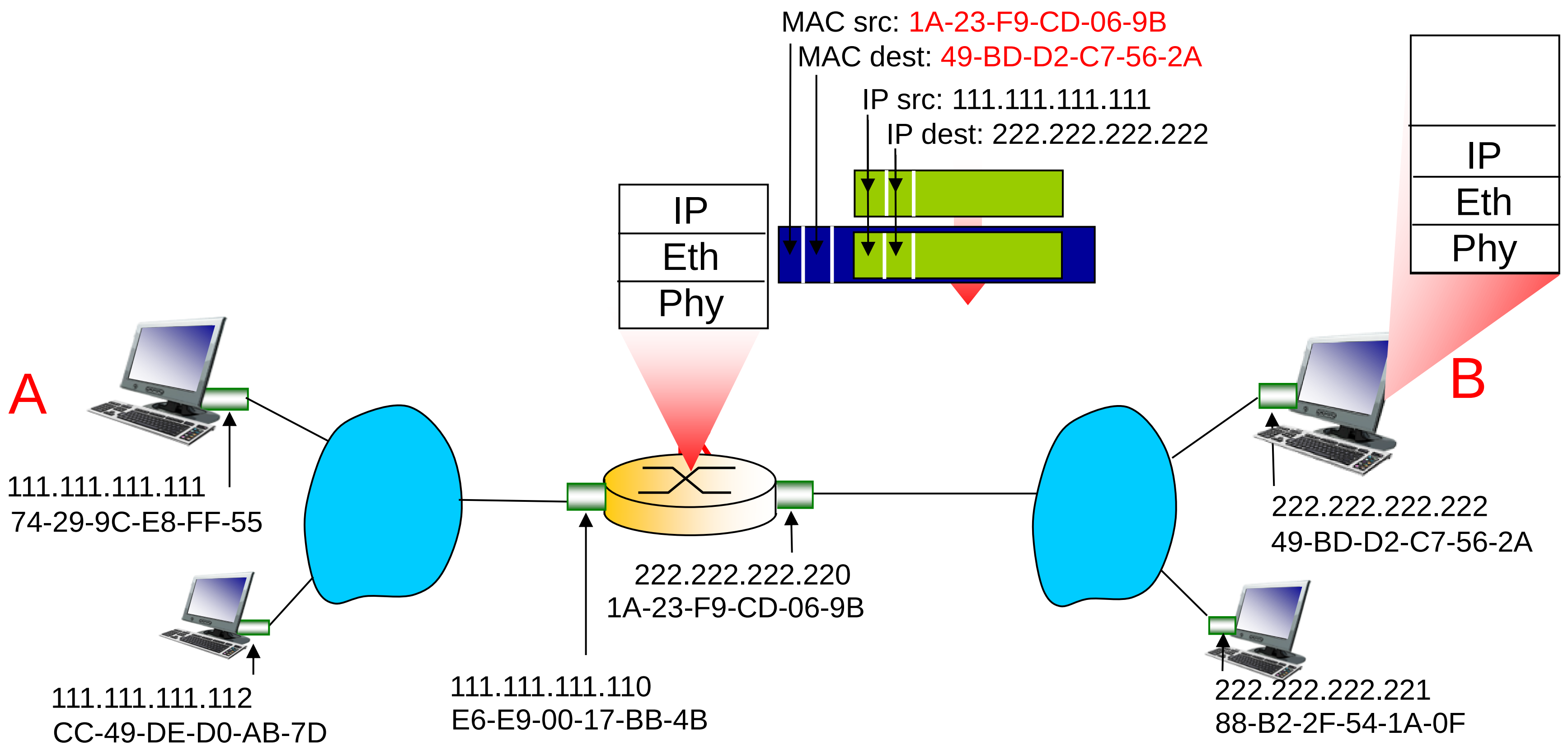
Addressing: routing to another LAN

- 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



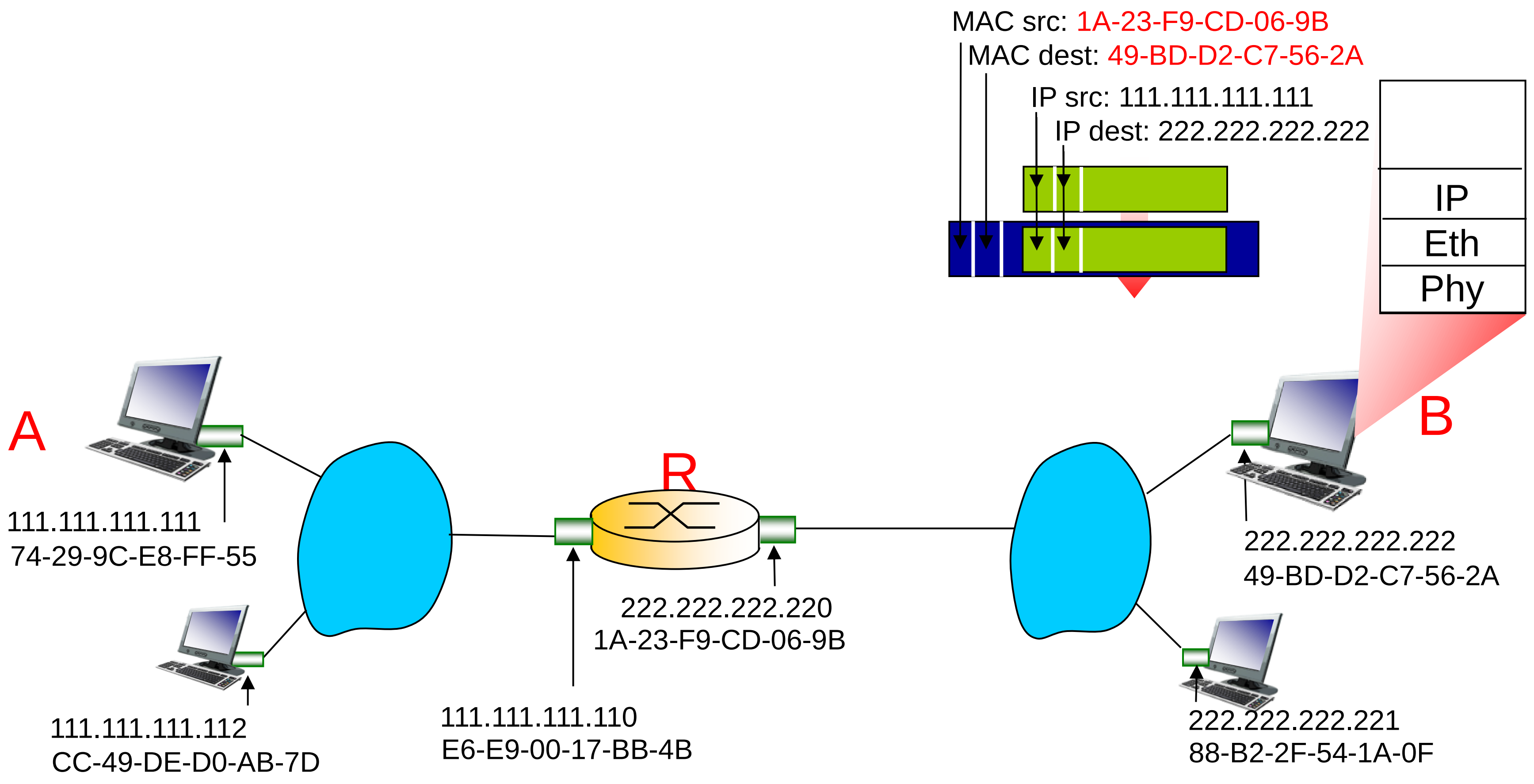
Addressing: routing to another LAN

- 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

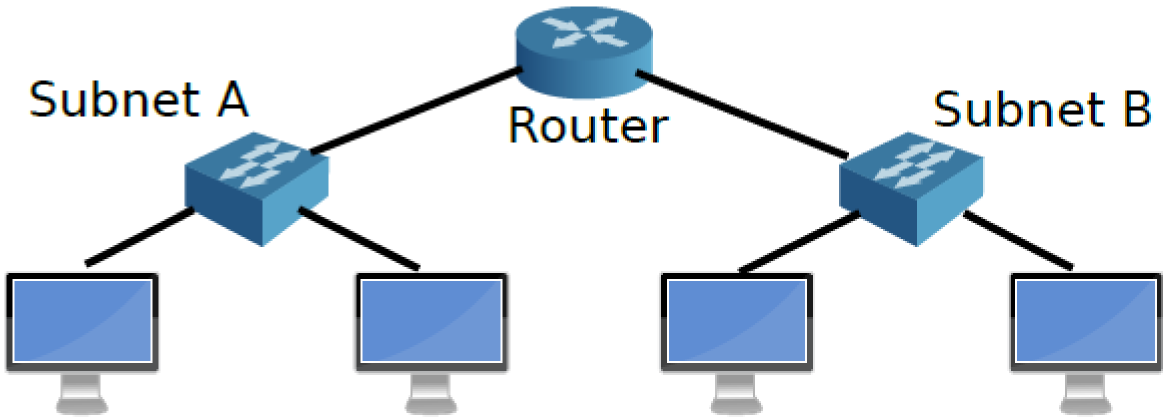


Hubs/Switches/Routers

- Hubs (not really used any more):
  - Layer 1 repeaters, no own MAC address
  - Re-transmit received packets on all ports
- Layer 2 Switches:
  - Passively learns MAC addresses from traffic
  - Store these in a CAM table (no active ARP)
  - Forward incoming packets to correct port (if known)
  - Hubs/Layer 2 switches are invisible to other network members
- Layer 3 Routers:
  - Router itself has MAC/IP address
  - Sender path would have router MAC as first hop
  - Router itself uses ARP/ MAC table like any host

Link-layer Broadcast Domain (LLBD)

- LLBD spans all *single-hop destinations*
  - L1 Hubs/ L2 Switches do not count as hop
  - Usually use one Network-layer subnet (>1 possible)
  - Network-layer subnets can only have one LLBD
  - Security/usability constraints
    - No filtering through firewall
    - Locally broadcasted services (printer, shared music, etc)
- Intuition: *physical proximity* implies *logical proximity*
- How to separate neighbors into two broadcast domains?



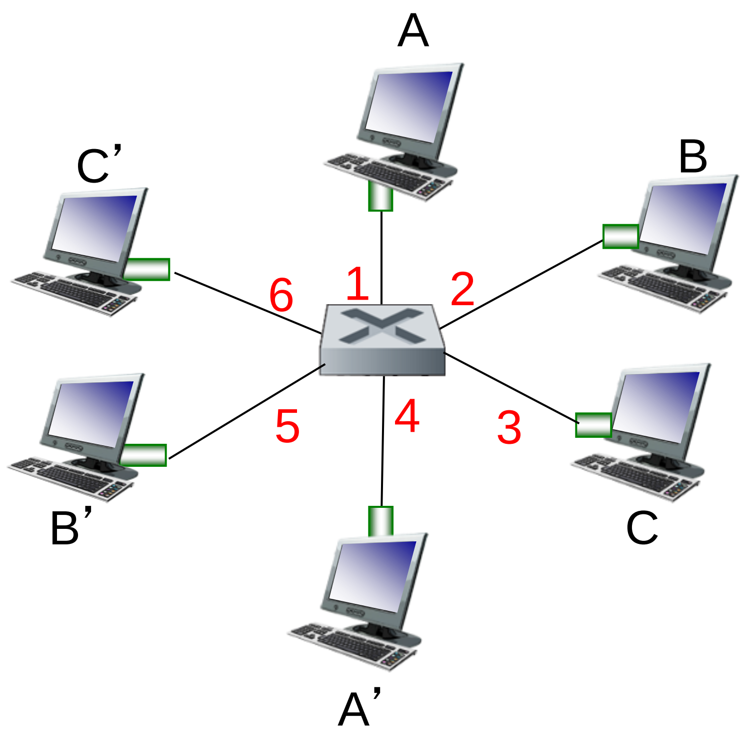
Two subnets connected by a router

Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Switch: *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' can transmit simultaneously, without collisions



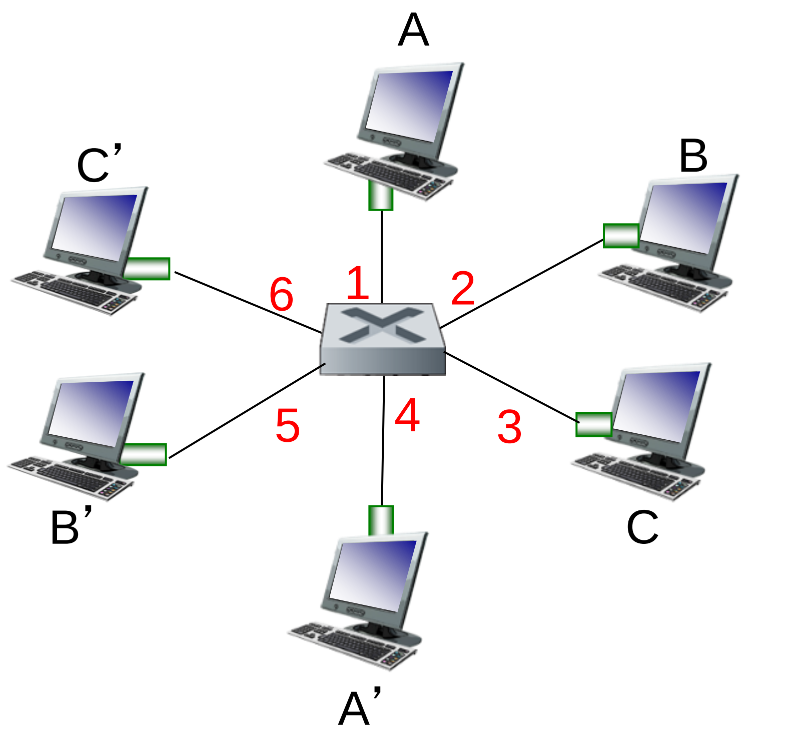
switch with six interfaces (1,2,3,4,5,6)

Switch forwarding table

Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

- A: each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
  - looks like a routing table!

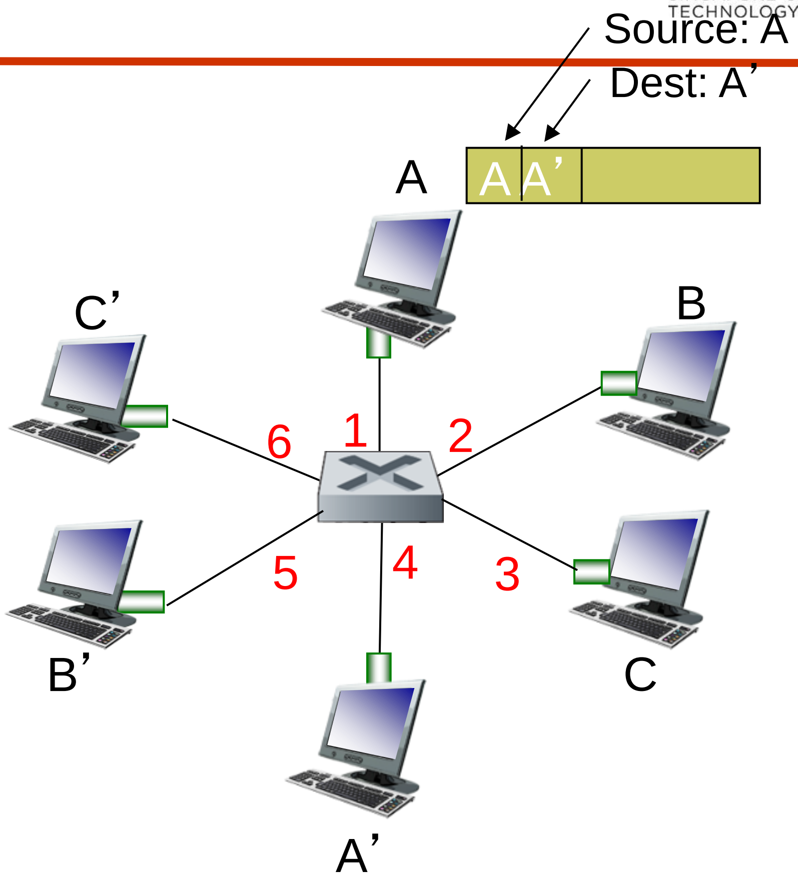
Q: how are entries created, maintained in switch table?
- something like a routing protocol?



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

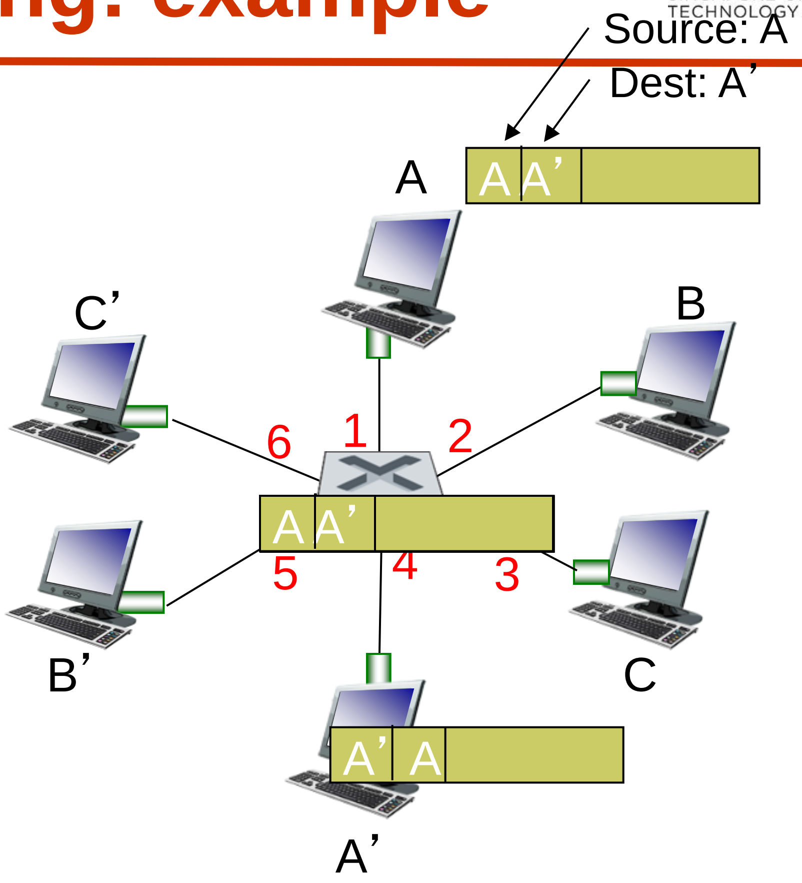
Switch: frame filtering/forwarding

```
when  frame received at switch:

    1. record incoming link, MAC address of sending host
    2. index switch table using MAC destination address
    3. if entry found for destination
          then {
         if destination on segment from which frame arrived
              then drop frame
              else forward frame on interface indicated by entry
           }
         else flood  /* forward on all interfaces except arriving
                        interface */
```

Self-learning, forwarding: example

- frame destination, A', locaton unknown: *flood*
- destination A location known: selectively send on just one link

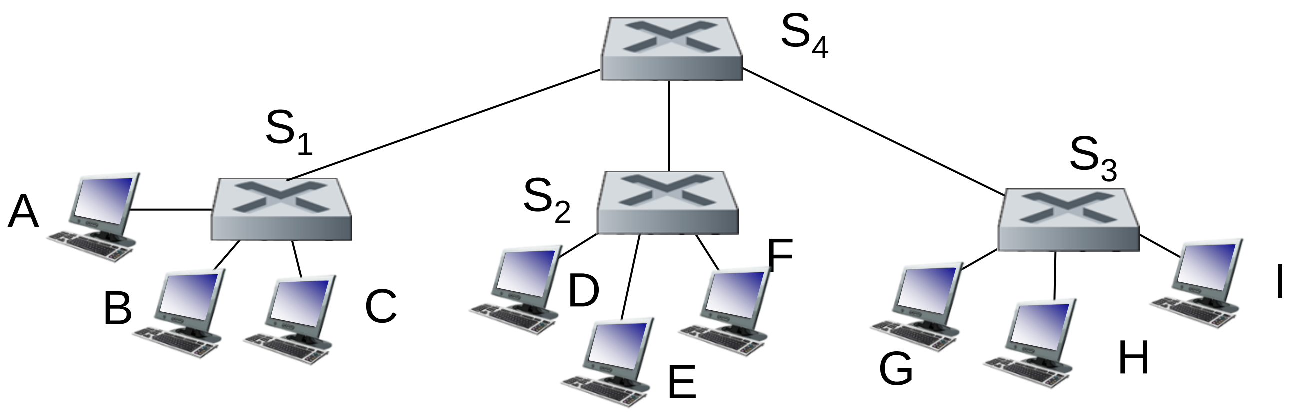


MAC addr	interface	TTL
A	*1*	*60*
A'	*4*	*60*

switch table (initially empty)

Interconnecting switches

- switches can be connected together

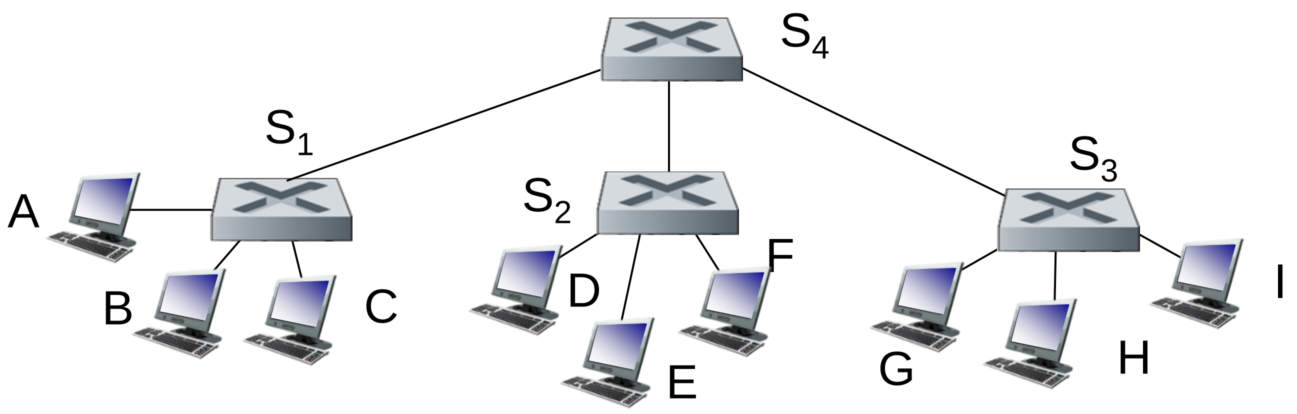


Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?

- A: self learning! (works exactly the same as in single-switch case!)

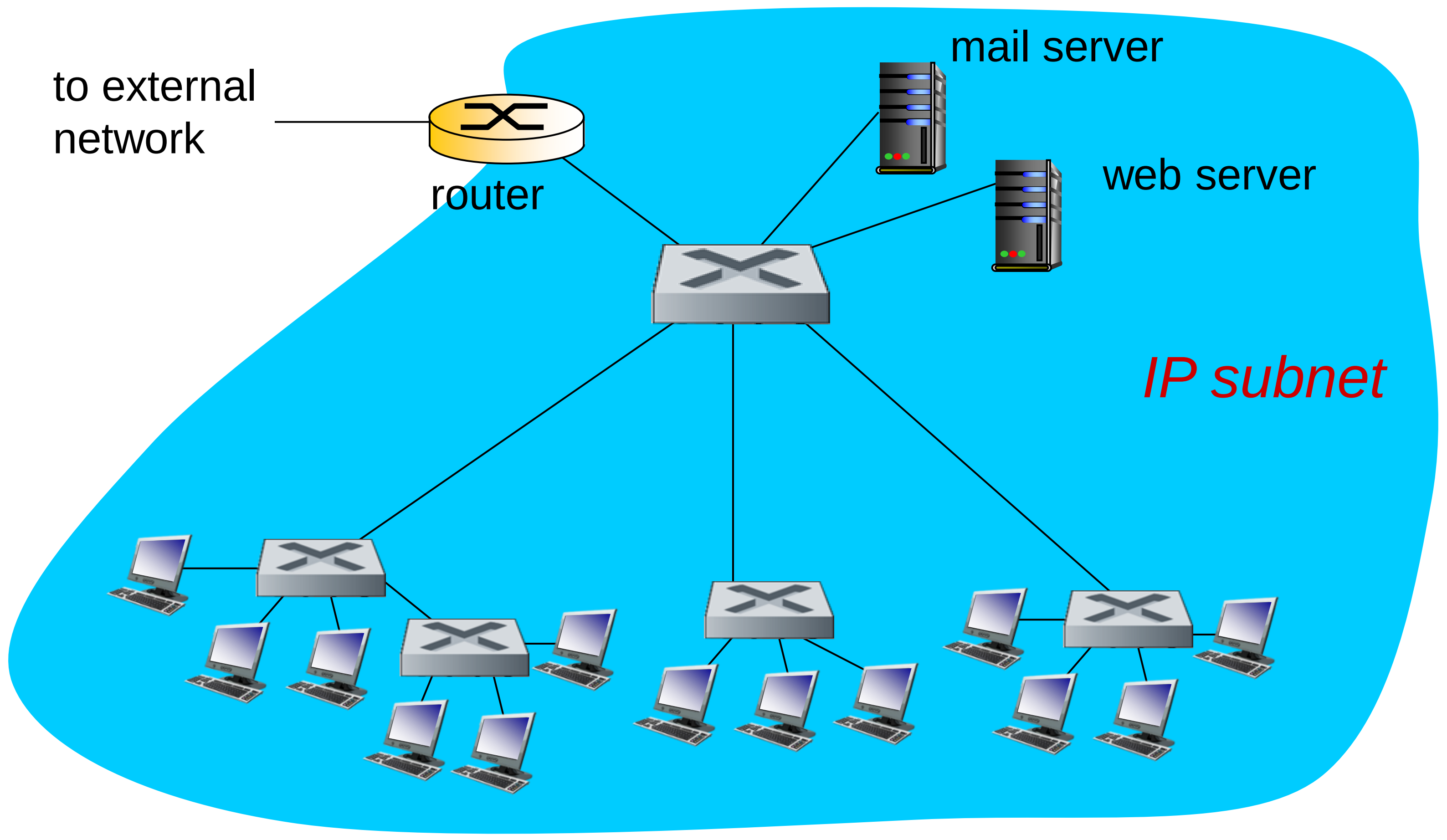
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Institutional network



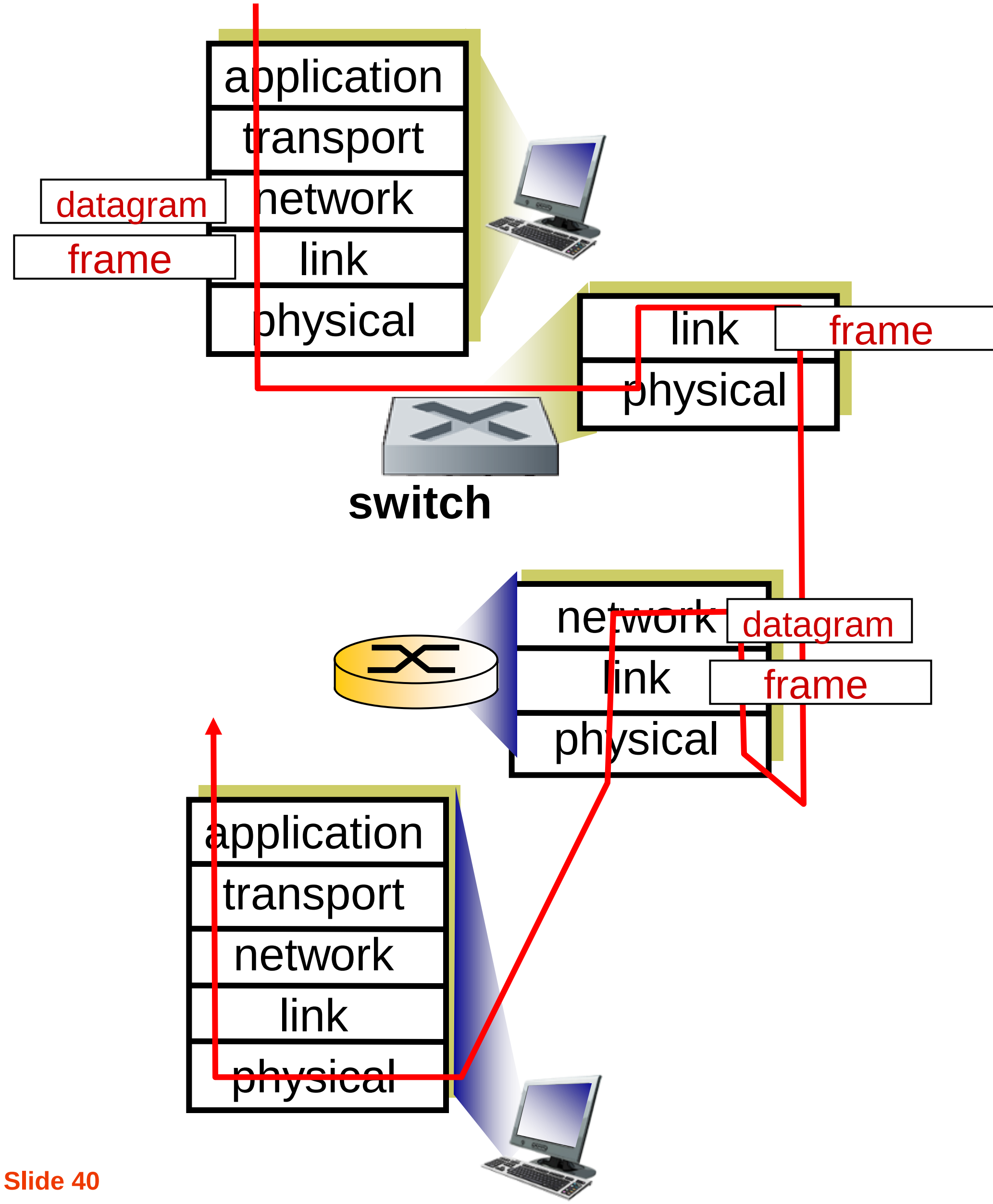
Switches vs. routers

both are store-and-forward:

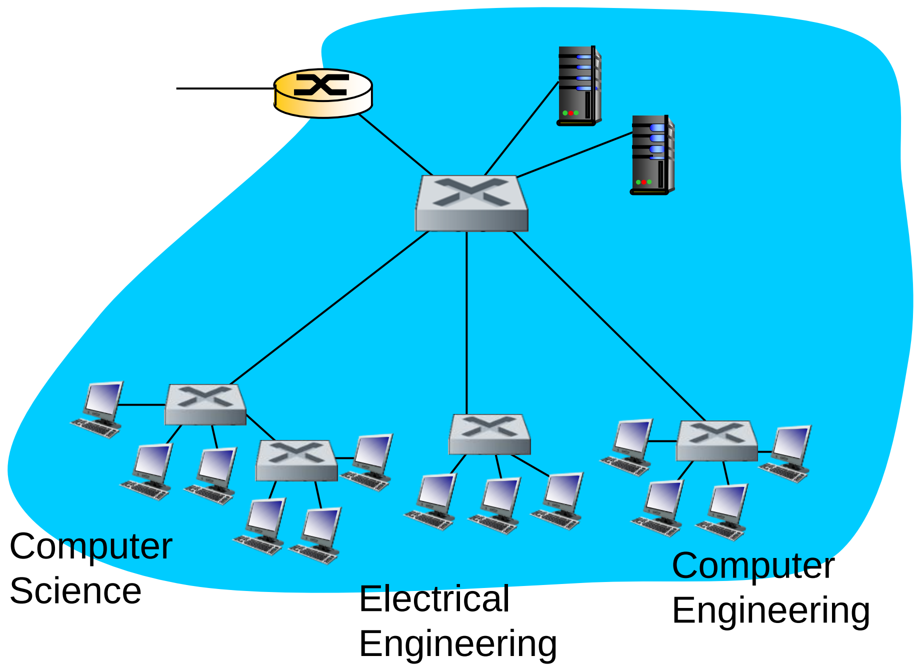
- *routers:* network-layer devices (examine network-layer headers)
- *switches:* link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



VLANs: motivation



consider:

- CS user moves office to EE, but wants connect to CS switch?
- single broadcast domain:
  - all layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - security/privacy, efficiency issues

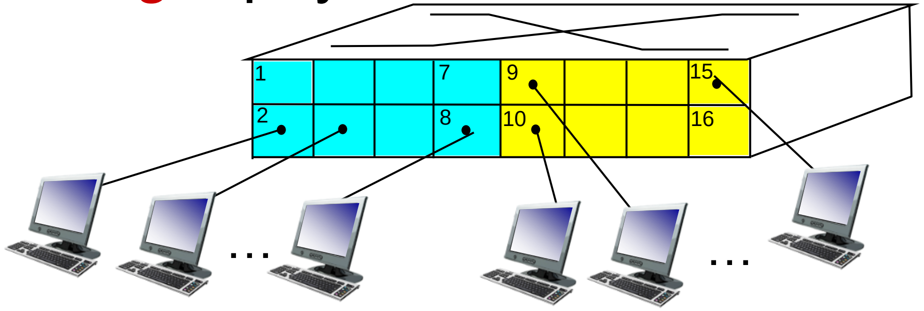
VLANs

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Virtual Local Area Network

switch(es) supporting VLAN capabilities can be configured to define multiple ***virtual*** LANS over single physical LAN infrastructure.

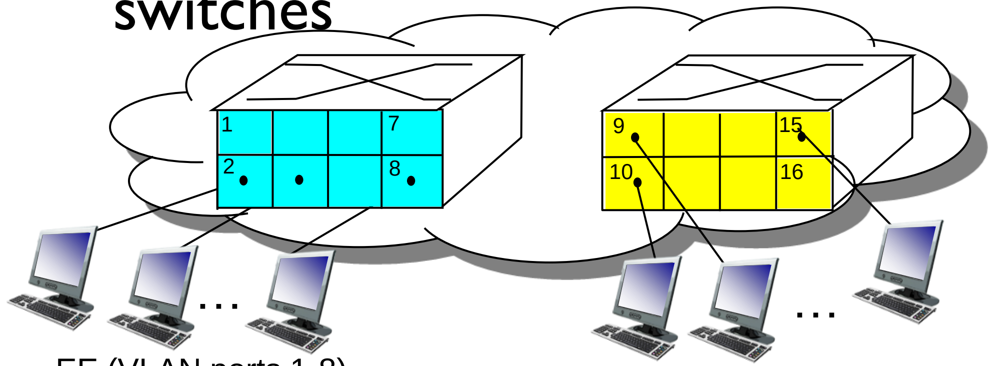
port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch ……



Electrical Engineering (VLAN ports 1-8)

Computer Science (VLAN ports 9-15)

… operates as *multiple* virtual switches

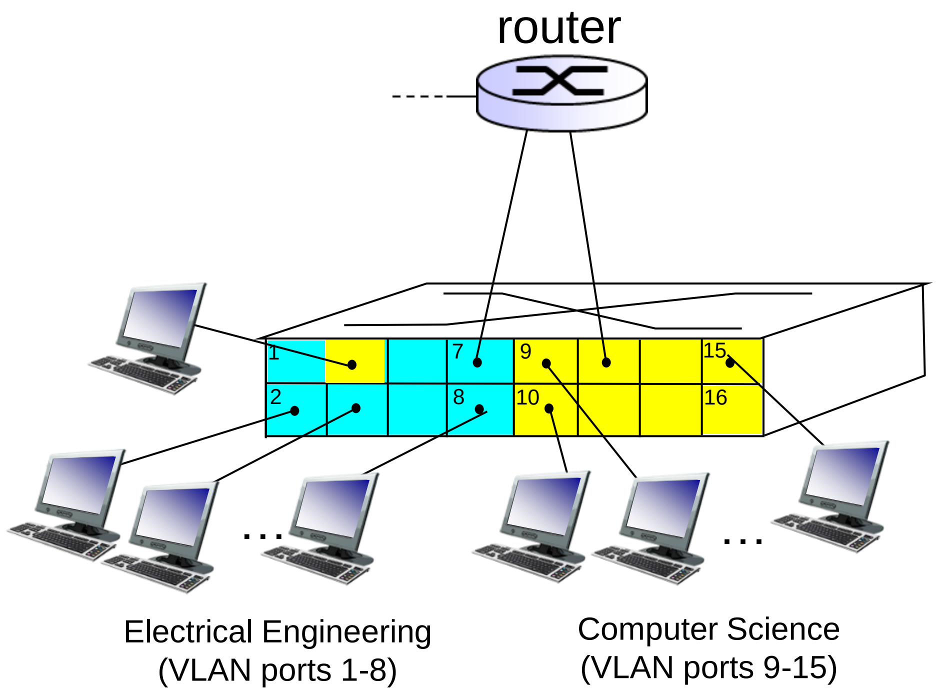


EE (VLAN ports 1-8)

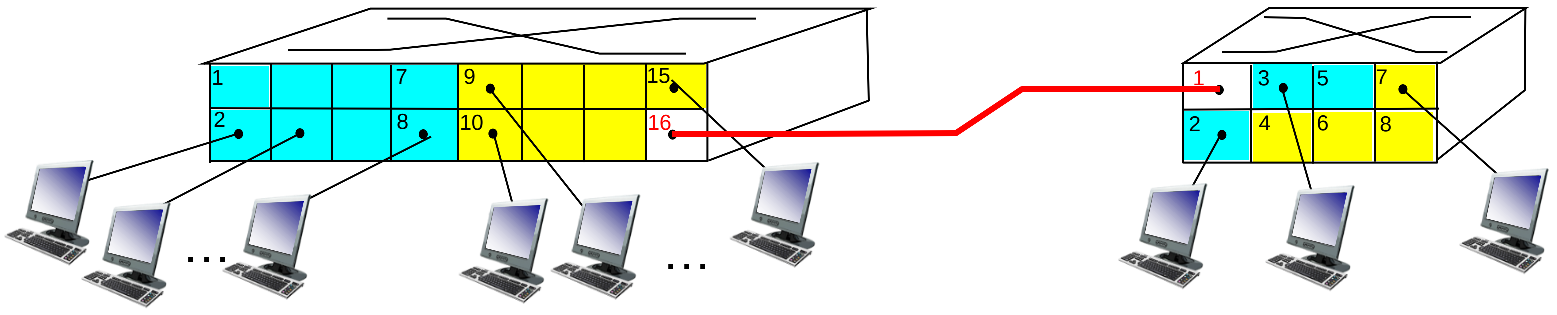
CS (VLAN ports 9-16)

Port-based VLAN

- *traffic isolation:* frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port
- dynamic membership: ports can be dynamically assigned among VLANs
- forwarding between VLANS: done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers



VLANS spanning multiple switches



Electrical Engineering
(VLAN ports 1-8)

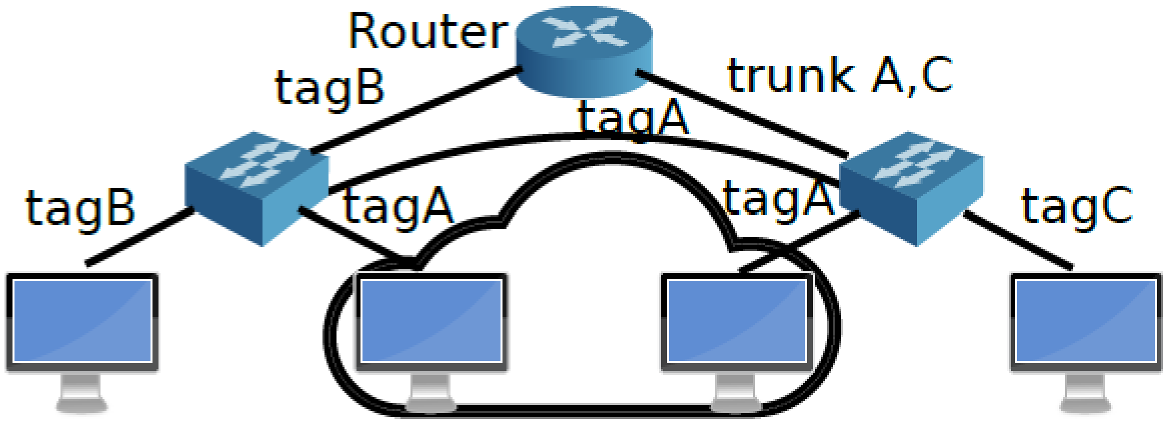
Computer Science
(VLAN ports 9-15)

Ports 2,3,5 belong to EE VLAN
Ports 4,6,7,8 belong to CS VLAN

- *trunk port:* carries frames between VLANS defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

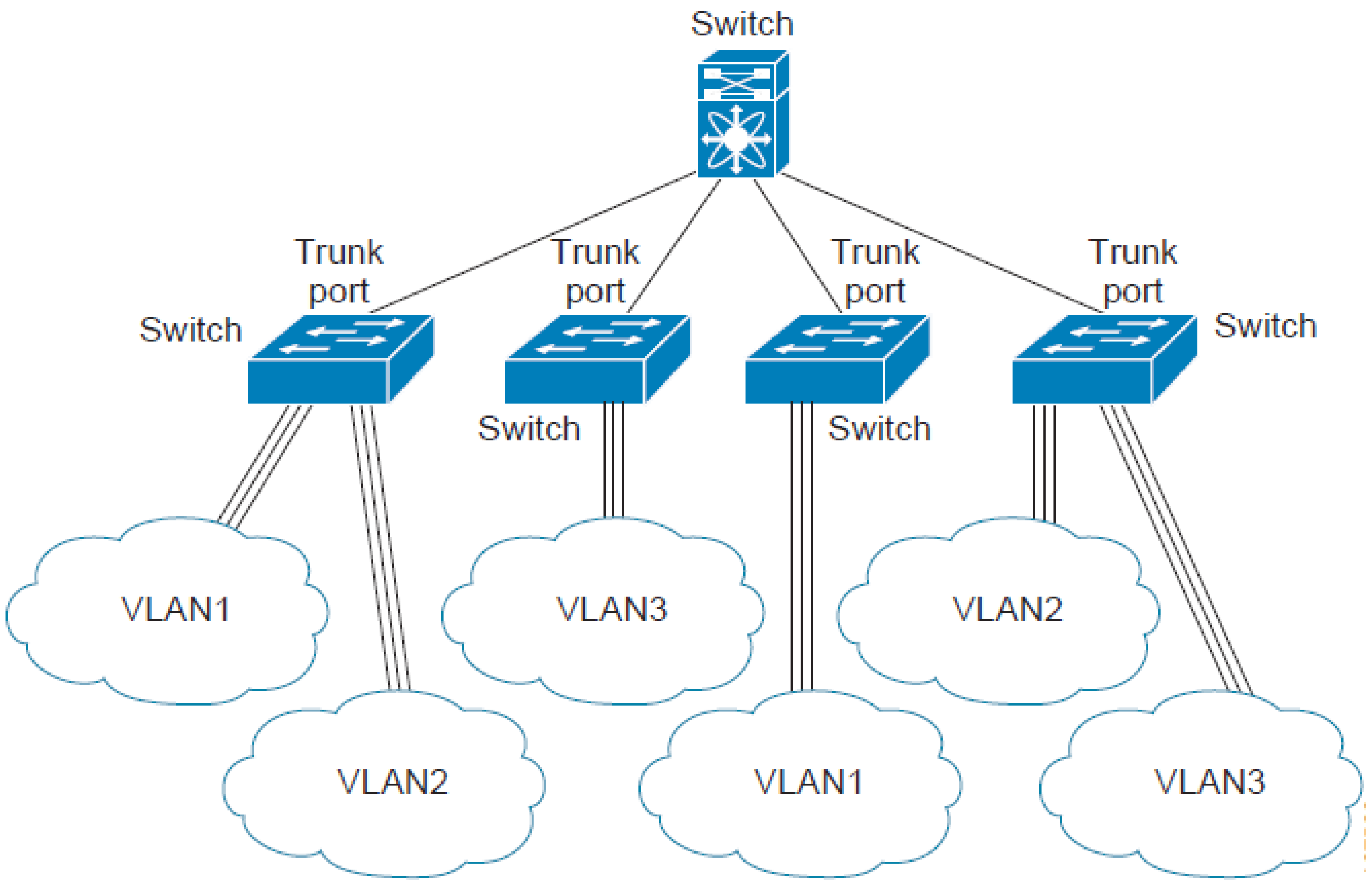
VLAN trunks

- One cable per VLAN is clearly inefficient
- How to send traffic of several VLANs over one cable?
- A VLAN ID is added in between Link and Network layer
- Defined by 802.1Q – additional 12 bit VID field after ETH header
- The sender adds VLAN tag, receiver removes it (or passes on)
- The sender needs to expect specific VLAN IDs



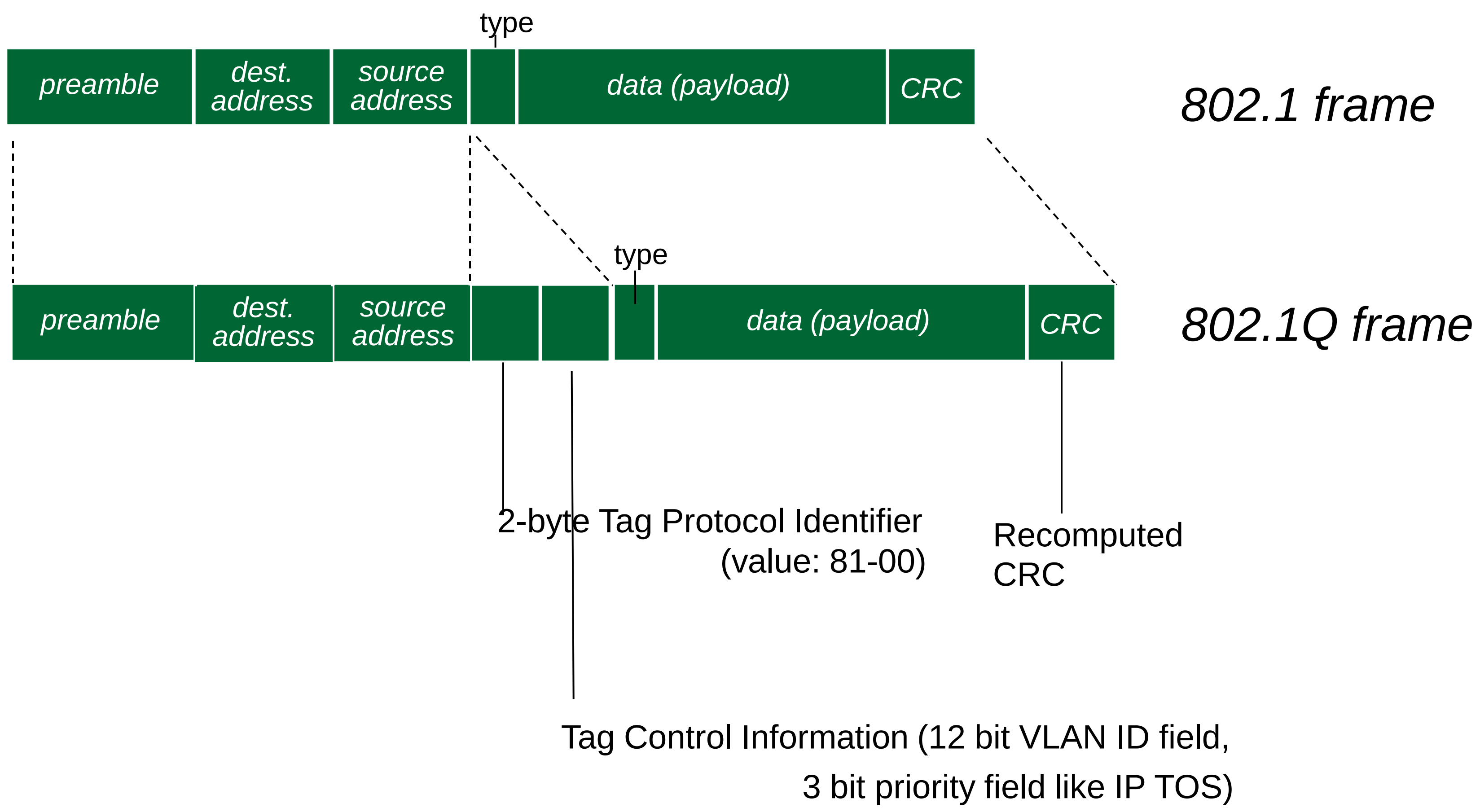
Note: In this configuration, A1 + A2 reach each other and the router

Devices in a trunking environment



https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/CLIConfigurationGuide/VLANs.html?referring_site=RE&pos=1&page=https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/configuration/guide/cli/CLIConfigurationGuide/AccessTrunk.html

802.1Q VLAN frame format



Activity 9: Switch table and VLANs (1)

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

A switch has three interfaces, labelled 1 to 3, and an empty switch or CAM table (initially). The switch receives four frames consecutively in the order listed in the table below. The source, destination, and the interface in which each frame is received is summarized in the table below. The action column in the table above refers to the action that the switch takes on each frame. Complete the table by filling the action column. An action can be either “forward to x” (x is one interface), or “broadcast”.

Incoming Interf	Source MAC	Destination MAC	Action taken
1	AA::AA::AA::AA::AA::AA	DD::DD::DD::DD::DD::DD	
1	AA::AA::AA::AA::AA::AA	BB::BB::BB::BB::BB::BB	
2	BB::BB::BB::BB::BB::BB	AA::AA::AA::AA::AA::AA	
3	DD::DD::DD::DD::DD::DD	BB::BB::BB::BB::BB::BB	

Activity 9: Switch table and VLANs (2)

Assume the next frame received is the following:

Incoming Interf	Source MAC	Destination MAC	Action taken
1	CC::CC::CC::CC::CC::CC	AA::AA::AA::AA::AA::AA	

By default, the switch will drop that frame. Please comment on why that is reasonable (or not).

Activity 9: Switch table and VLANs (3)

Assume that the switch mentioned above is now configured with two VLANs. Port 1 is configured as Trunk port for VLAN1 and VLAN2, while port 2 is an Access port for VLAN1, and port 3 is an Access port for VLAN2. For each of the incoming and outgoing frames in the following table, please state what the VLAN ID (VID) in the header should be to ensure that frame 3 and 4 can reach their destination. If the frame should not contain an 802.1q header, please put VID as "None".

Inc. Int.	Source MAC	Destination MAC	Incoming VID	Outgoing VID
1	AA::AA::AA::AA::AA::AA	DD::DD::DD::DD::DD::DD		
1	AA::AA::AA::AA::AA::AA	BB::BB::BB::BB::BB::BB		
2	BB::BB::BB::BB::BB::BB	AA::AA::AA::AA::AA::AA		
3	DD::DD::DD::DD::DD::DD	AA::AA::AA::AA::AA::AA		

Conclusion

- We discussed the main tasks of the link layer:
  - Interface addressing
  - Transmission medium specific checksum
  - Local subnet creation
    - Used for broadcasts, security, logical separation