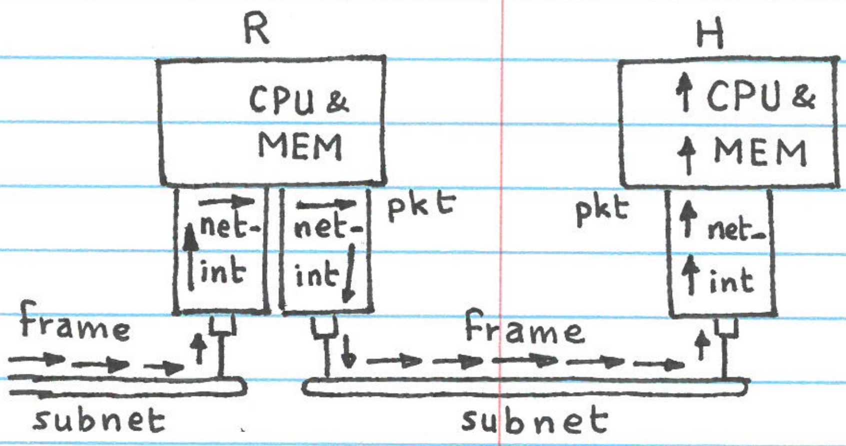
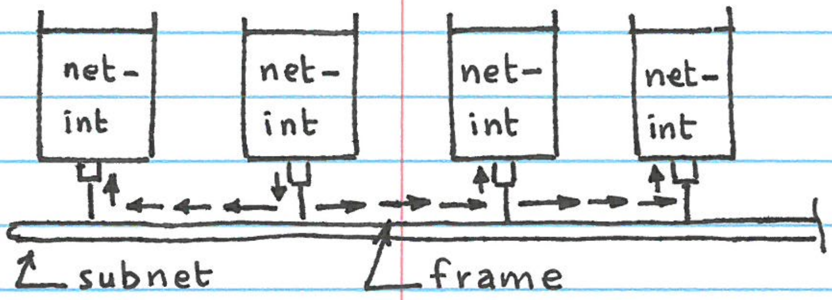
Link Layer Protocol



Notation:
net-int:
network-interface

- the main function of the link layer protocol is to rcv a frame from one subnet and forward the frame to another subnet towards the dst of the pkt contained in the frame



- when a net-int sends a frame on a subnet, the frame is sent to every other net-int attached to the subnet

- each net-int has a unique world-wide MAC address

- each frame (sent on a subnet) has a header. This header contains the dst MAC address of the net-int that should rcv the frame after it is sent on the subnet

IP Addresses of Net_ints

- each net_int has a unique IP address
- the IP address consists of 4 Bytes:

$$w.\underbrace{x}_{\text{integer in range } [0..255]}.y.z$$

- broadcast IP address is 255.255.255.255
- IP addresses can be changed by software and can be used in routing IP pkts between routers

MAC Addresses of Net_ints

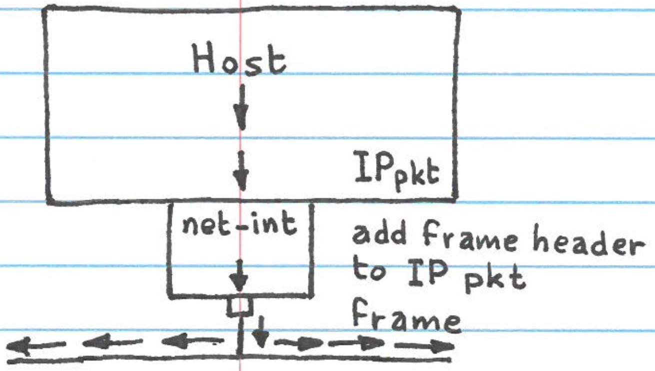
- each net_int has a unique MAC address
- the MAC address consists of 6 Bytes. Each byte is written as 2 hexadecimal values in the range 0..F
- broadcast MAC address is FF_FF_..._FF
- MAC addresses are fixed in hardware and can be used in routing link frames between switches in switched Ethernets

Frame Headers

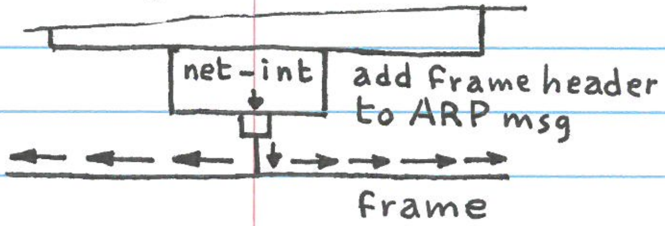
- a frame header has 5 fields:

1. src MAC address	6 Bytes
2. dst MAC address	6 Bytes
3. type of data in Data Field	2 Bytes
4. Data is either an IP pkt or or an ARP (query or response) msg	
5. Cyclic Redundancy Check (CRC)	4 Bytes

- Data is IP pkt:



- Data is ARP msg:



Detecting Corruption in Frames

- before a net_int sends a frame over a subnet, the net_int computes the expected value of the CRC field in the frame header

- when a net_int rcvs a frame from a subnet, it (1) computes the expected value of the CRC field, (2) compares this expected value with the rcvd value of the CRC field, and (3) concludes that the rcvd frame is corrupted (and should be discarded) iff the expected and rcvd values do not match.

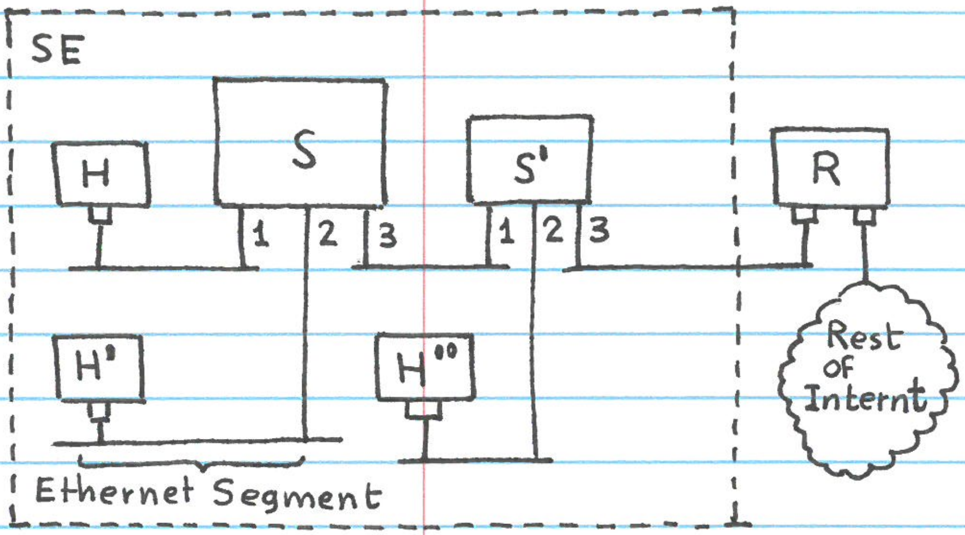
Architecture of Link Layer

- each link layer has one subnet. Because subnets come in different technologies (e.g. switched Ethernets, wireless Ethernets, phone lines, TV cables, and satellite links), the architecture of a link layer depends on the architecture of its subnet.

- from now on, we focus on link layers whose subnels are Switched Ethernets (SEs)

Architecture of Switched Ethernets

- example:

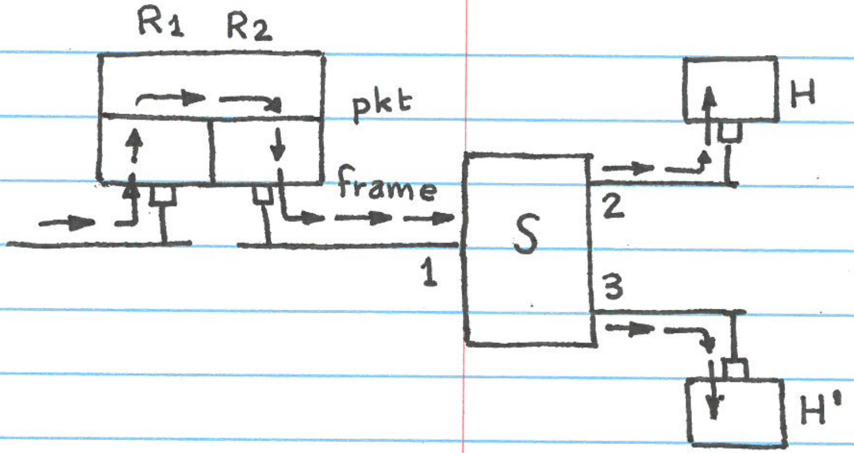


- notation: SE : Switched Ethernet
  S, S': switches
  H, H', H'': hosts
  R : router

- topology of SE is a tree

- attached to each Ethernet segment are two computers, at least one of them is switch, and at most one of them is host or router

Resolving IP Addresses into MAC Addresses



- for router R2 to send an IP pkt to a host whose IP address is H, R2 needs to encapsulate the pkt in a frame whose src MAC address is the MAC address of R2, and whose dst MAC address is the MAC address of H

- Problem:
  How does R2 know the MAC address of H?

- Solution:
  Use 2 steps of Address Resolution Protocol (ARP)

Step 1 of ARP

- R_2 broadcasts an ARP query encapsulated in a frame to all computers attached to SE to which R_2 and H are attached

- header of encapsulating frame:

src MAC address	= MAC address of R_2
dst MAC address	= FF_..._FF (broadcast MAC address)
Data	= ARP query asking for the MAC address of the computer whose IP address is H

Step 2 of ARP

- each computer attached to the SE rcvs a copy of the ARP query that R2 broadcasted, but only the computer whose IP address is H gets to reply by sending back an ARP response to R2

- the ARP response is encapsulated in a frame whose header is as follows:

src MAC address	= MAC address of H
dst MAC address	= MAC address of R2
Data	= ARP response identifying the MAC address of the computer whose IP address is H

ARP Table

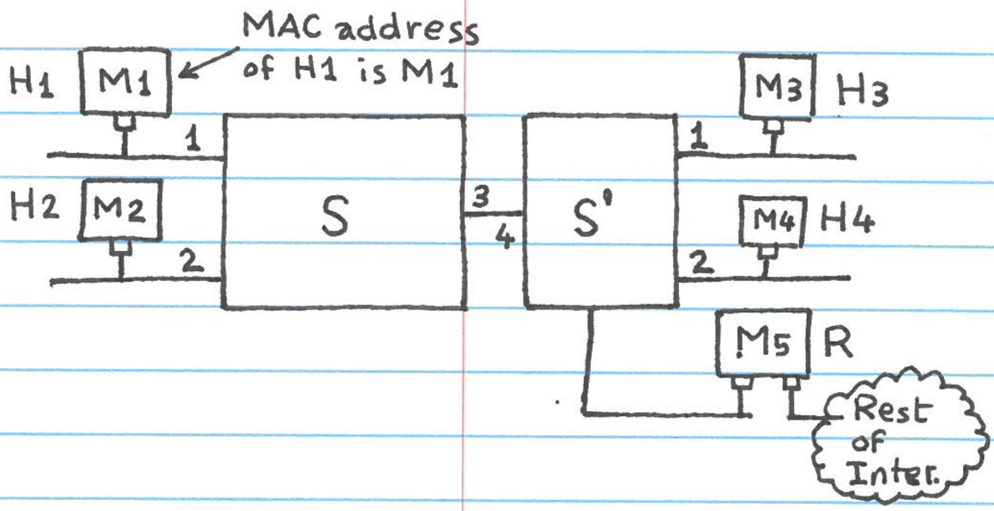
- after R2 resolves an IP address into its corresponding MAC address, R2 stores the following entry in its ARP table:
  (IP address, corresponding MAC add., TTL)
  The initial value of TTL is 20 minutes

- until the value of TTL in this entry expires, R2 uses this entry to resolve the IP address into its corresponding MAC address

- when the value of TTL in this entry expires, R2 uses ARP to resolve the IP address into its corresponding MAC address, and the cycle repeats

Routing Frames in SEs

- each switch in SE has a switching table:



- switching table of switch S in the above SE:

dst MAC addr.	interface of S to reach dst MAC add.	TTL
M1	1	
M2	2	
M3	3	
M4	3	
M5	3	

- initially, the switching tables of all switches are all empty

Scenario to Fill Switching Tables of S and S'

- refer to previous slide ...

- if host H2 sends a frame (src=M2, dst=M4) over interface 2 of S, then entry (M2, 2, maxTTL) is added to table of S and frame (M2, M4) is broadcasted to interfaces 1 and 3 of S

- frame (M2, M4) to interface 1 is discarded by host H1, and frame (M2, M4) to inteface 3 is rcvd by S', then entry (M2, 4, max TTL) is added to table of S', and frame (M2, M4) is broadcasted to interfaces 1, 2, and 3 of S'

- frame (M2, M4) to interfaces 1 and 3 are discarded, and frame (M2, M4) to interface 2 is rcvd by its final dst host H4

- when TTL in any entry of a switching table becomes 0, then this entry is discarded from the table.

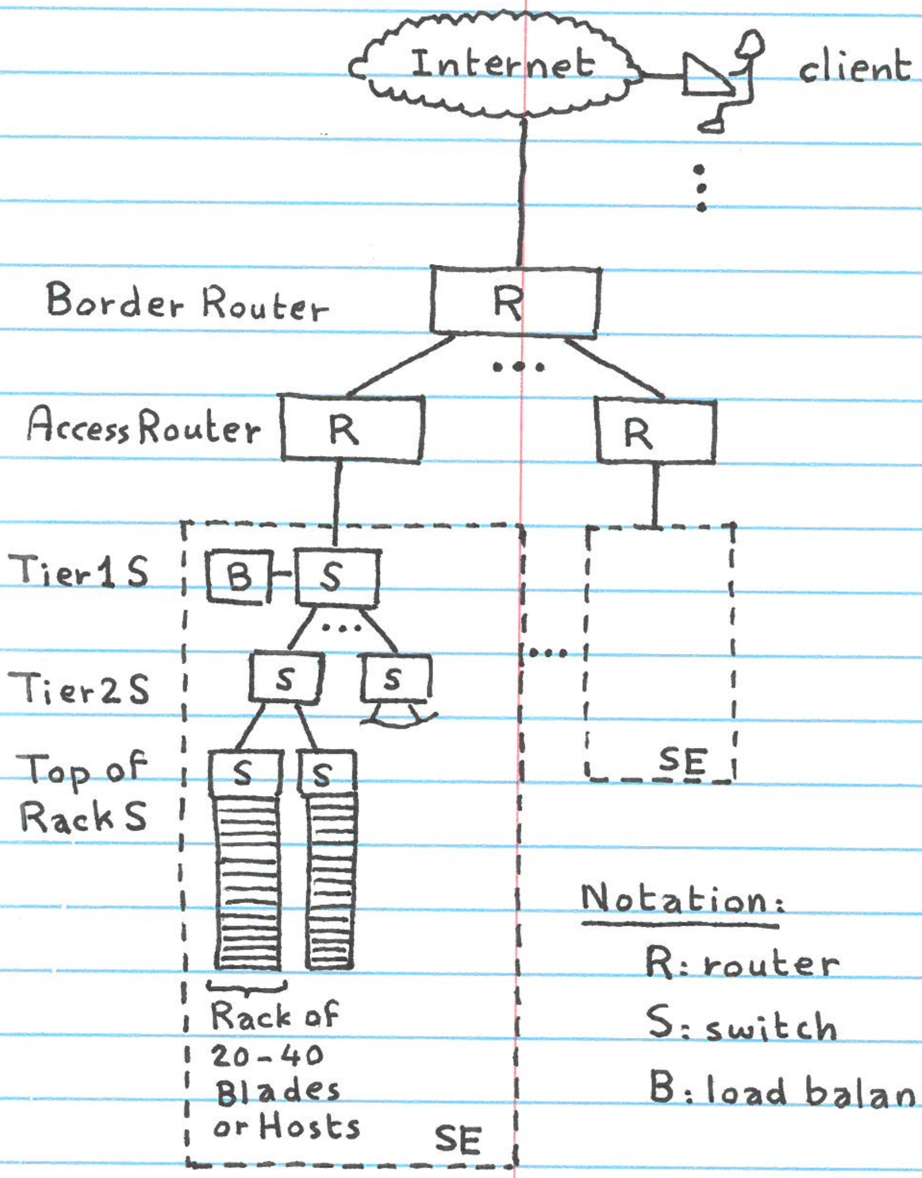
Data Centers

- each Internet company (e.g. Google, Microsoft, ..) has a massive data center with about 10^5 hosts to support processing distinct applications, such as web, email, search, in the company

- the building block of a data center is SEs

- the architecture of each data center is a trade secret but can be roughly represented as follows :

Architecture of Data Centers



Notation:

R: router

S: switch

B: load balancer

- each application, that is supported by the data center, is assigned a well-known public IP address

- any client C, that needs a service from an application App, sends a request Rq to the public IP address of App, and later rcvs response packets from the public IP address of App as follows

NAT_like Operation of Load Balancers

1. when client C sends a request Rq, whose src is C and whose dst is the public IP address of application App, Rq is directed to a load balancer B for App

2. when B rcvs Rq, it performs 4 tasks: (1) B modifies src of Rq to become B, (2) B modifies dst of Rq to become a private host H (known only to B and is capable of performing the request), (3) B forward modified Rq to H, and (4) B enters following entry to its NAT table

$$\begin{tikzcd}
\boxed{\begin{array}{l} \text{src} = H \\ \hline \text{dst} = B \end{array}} \arrow[r, "\text{become}"] & \boxed{\begin{array}{l} \text{src} = \text{public IP addr.} \\ \hline \text{dst} = C \end{array}}
\end{tikzcd}$$

3. when B rcvs reply Rp, it uses its NAT table to update the src and dst of Rp. Finally B forwards the modified Rp to C