REVIEW OF INTERNETWORKING

Lecture 2 Link Layer

1. Basic introduction

Data Link Layer has responsibility transferring datagram between adjacent nodes over a link. (Data link layer packet is a frame, encapsulates datagram into frame.)

Link addressing: MAC address

2. Collision in Ethernet:

这句话的意思是说：所有站共享媒体（信道），并且一次只能由一个站使用这个媒体。这也表示由某一个站所发送的帧将被所有站接收（广播方式）。只有真正的目的站才收下这个帧，而其他站都会丢弃。在这种情况下，我们怎样才能保证两个站不会在同一时间使用媒体呢？如果两个站同时使用媒体，它们发送的帧就会发生碰撞。

CSMA: Listen before sending. Half-duplex link. But it cannot avoid all collision because there is propagation delay of each frame.

CSMA/CD: Listen while sending. If collision is detected, abort transmission and retry.

站 A 传输的持续时间为 $t_4 - t_1$，而站 C 传输的持续时间为 $t_3 - t_2$。稍后我们将会看到，要使这个协议正常工作，任何一个帧的长度除以比特率都必须大于这两个持续时间的长度。在 t_4 时刻站 A 尚未完成的帧传输被放弃。同样在 t_3 时刻来自站 C 的尚未完成的帧传输被放弃。

帧：frame.

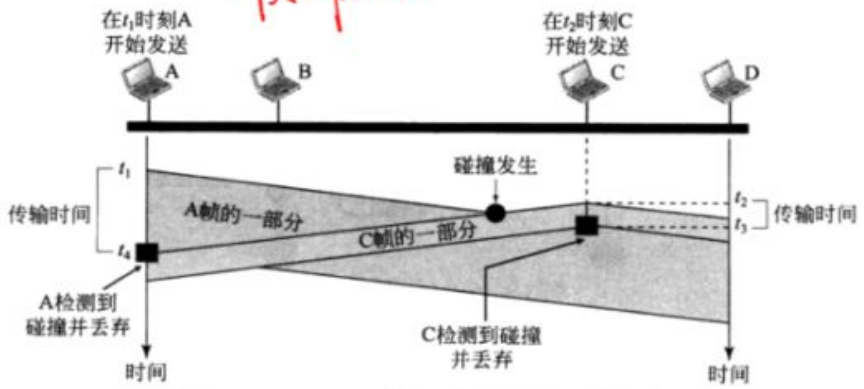


图 3.8 CSMA/CD 中第一个比特发生碰撞的情况

最小帧长

帧：frame

要使 CSMA/CD 正常工作，我们必须要限制帧的长度。如果某次传输发生了碰撞，那么正在发送数据的站必须在发送该帧的最后一比特之前放弃此次传输，因为一旦整个帧都被发送出去，那么该站将不会保留帧的复本，同时也不会继续监视是否发生了碰撞。因此帧的传输时间 T_{fr} 必须至少是最大传播时间 T_p 的两倍。要想了解其原因，让我们来设想一个最糟糕的场景。如果发生碰撞的两个站之间的距离最远，那么来自第一个站的信号需要花费 T_p 时间才能到达第二个站，而碰撞带来的影响又花了另一个 T_p 的时间才到达第一个站。因此，我们的要求就是第一个站必须在 $2T_p$ 长的时间后仍然在传输数据。

例 3.3

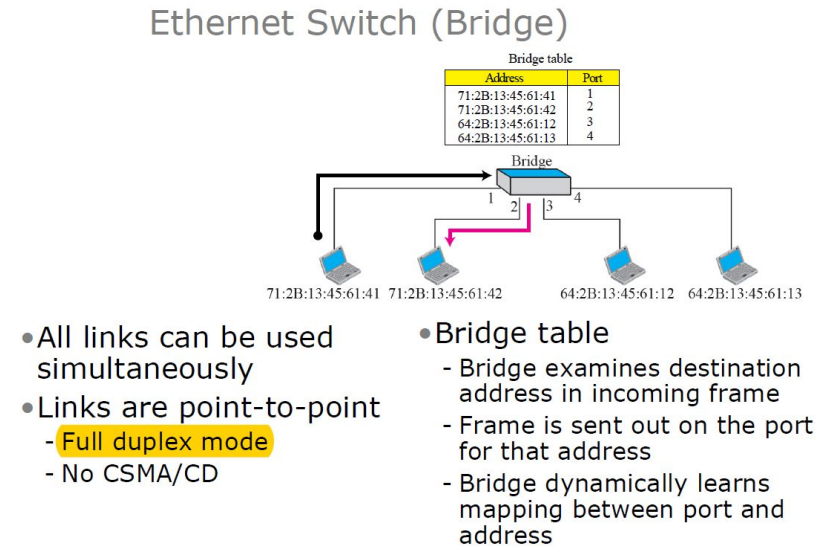
在标准以太网中，如果最大传播时间是 25.6μs，那么帧的最小长度是多少？

帧长/比特率

解

帧的传输时间是 $T_{fr} = 2 \times T_p = 51.2\mu s$。也就是说，以最坏的情况下，一个站需要经过长达 51.2μs 的传输时间才能检测到碰撞，因此这个帧的最小长度是 10 Mbps×51.2μs=512 比特或 64 字节。这实际上就是标准以太网中帧的最小长度，之前我们曾提到过。

No collision (nowadays used):



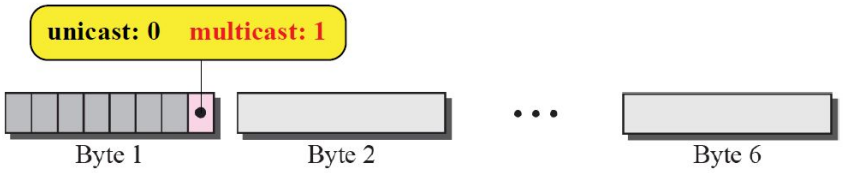
Bridge learns location of MAC addresses by inspecting (检查) source address field in incoming frames.

3. Ethernet (有线) address & MAC address

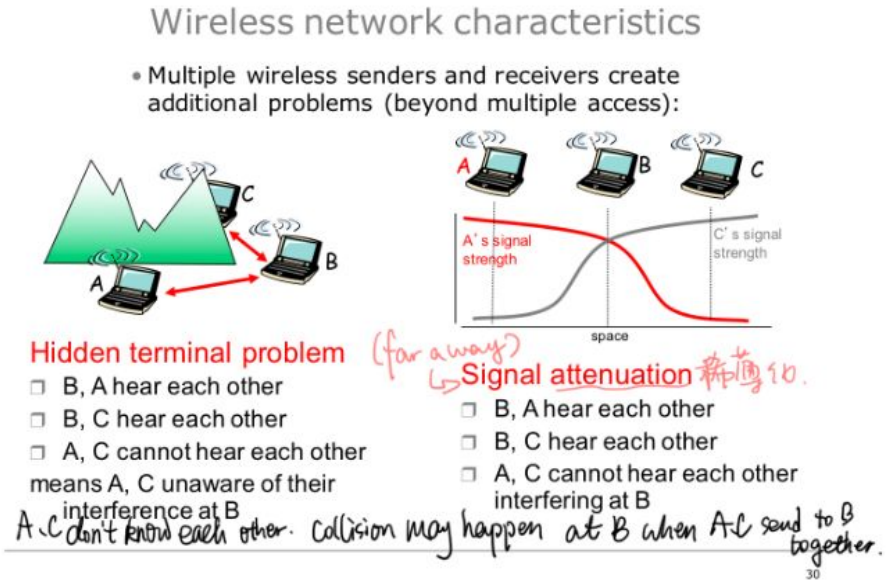
The both are 48-bit number used to uniquely identify each computer in a network. The address is usually written in hexadecimal form.

The Ehernet address is the most common form of MAC address.

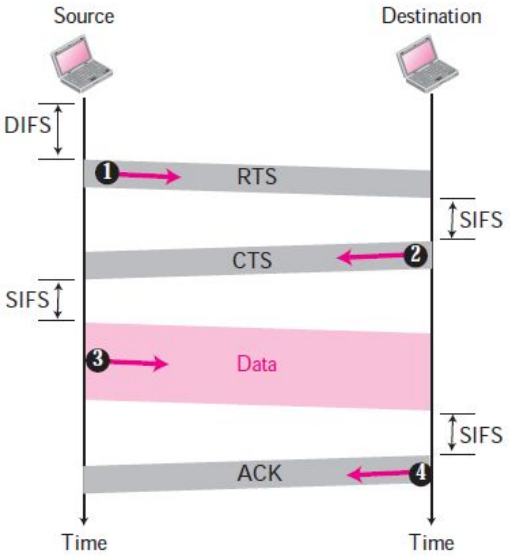
- All-ones is broadcast (multicast to all nodes)



4. Collision in Wireless LAN (Problem 1. Hidden station problem)



To avoid collision (Now we cannot use CDMA/CD), we use CDMA/CA:



1．源站在发送帧之前首先要检查载波频率上的能量值以检测媒体是否空闲。

a．源站使用带退避的坚持（persistence）策略等待信道空闲。

b．源站在发现信道空闲之后先等待一段称为**分布帧间距**（distributed interframe space，DIFS）的时间，然后再发送一个称为请求发送（RTS）的控制帧。

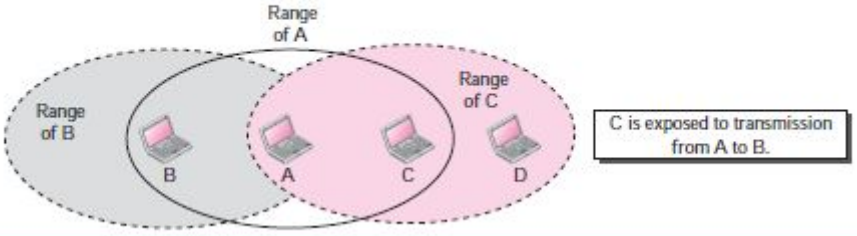
2．目的站在接收到这个 RTS 并等待了一段称为**短帧间距**（Short interframe space，SIFS）的时间之后，向源站发送一个称为允许发送（CTS）的控制帧，这个控制帧表示目的站准备好接收数据。

3．源站在等待了一段与 SIFS 等长的时间之后开始发送数据。

4．目的站在等待了与 SIFS 等长的时间之后发送确认帧，以表示该帧已接收。在这个协议中确认帧是很有必要的，因为源站没有任何其他手段可用于检查自己的数据是否成功到达了目的站。从另一方面说，在 CSMA/CD 中没有检测到碰撞这件事本身就是在告诉源站数据已到达目的站。

Problem 2 (RTS and CTS cannot solve this problem this time.)

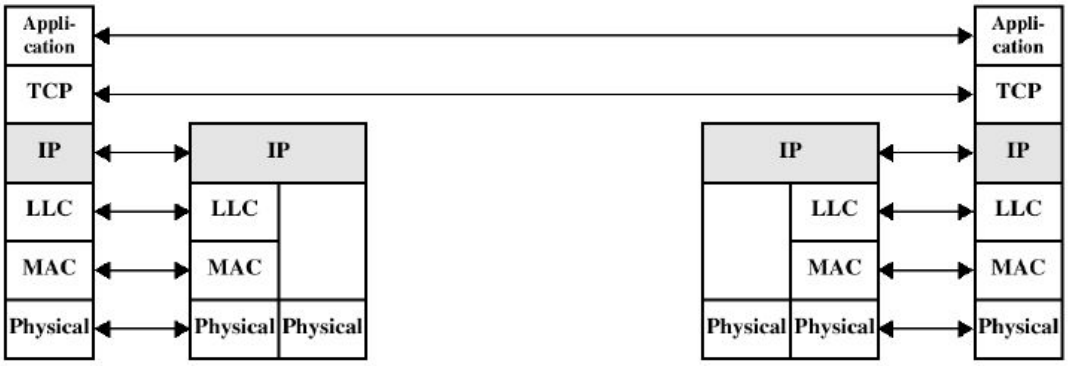
Figure 3.21 *Exposed station problem*



暴露站问题　现在要考虑一个与前面正好相反的情况，即暴露站问题。这个问题在于一个站可能会在信道实际上可用时却抑制了发送。在图 3.21 中，站 A 正在向站 B 传送数据，而站 C 有一些数据要发送给站 D，这些数据本来可以在不打断站 A 到站 B 的传输的情况下正常发送，但是因为站 C 暴露在站 A 的传输中，也就是说它能听到站 A 发送的数据，因而抑制了发送。换言之，站 C 因太过保守而浪费了信道容量。

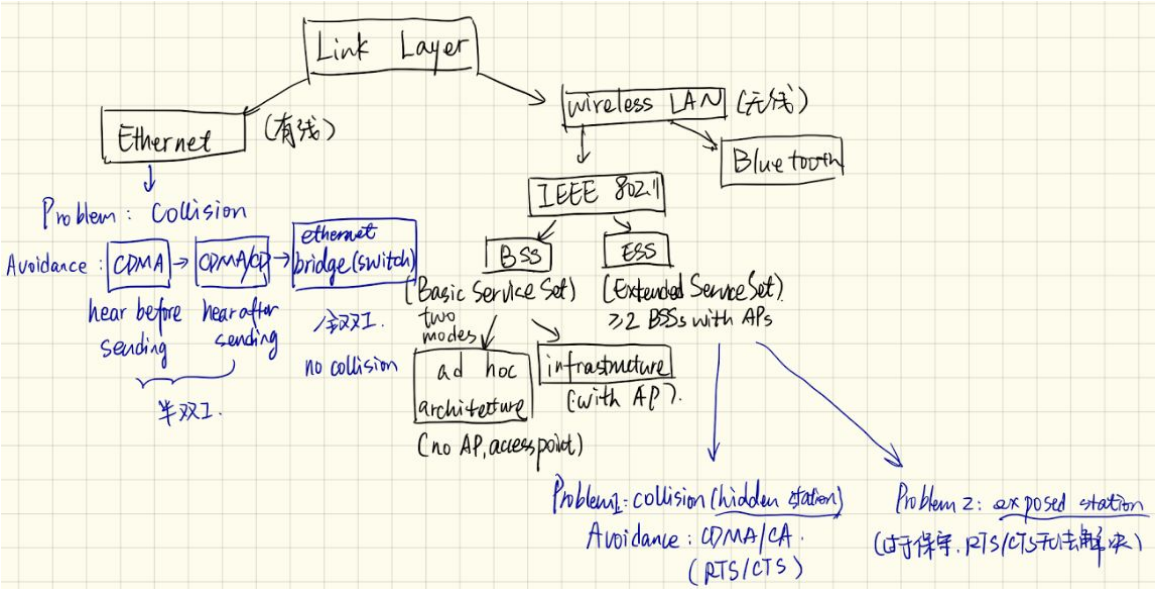
C have to wait to send data to D until A finish sending data to B.

5. 数据链路层还分MAC和LLC子层



MAC连接LLC子层和下级物理层。从LLC子层接受数据，附加上MAC地址和控制信息后再校验一下放到物理层去。
其中最重要的概念是MAC address。

Review of Lecture 2:



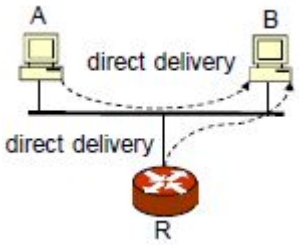
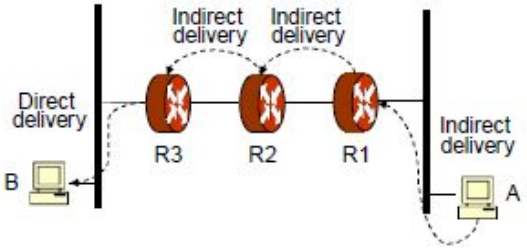
Lecture 3 Network layer fundamentals - Basic forwarding IP addressing

1.In general

End-to-end delivery of packets independent of the underlying link layer technologies.

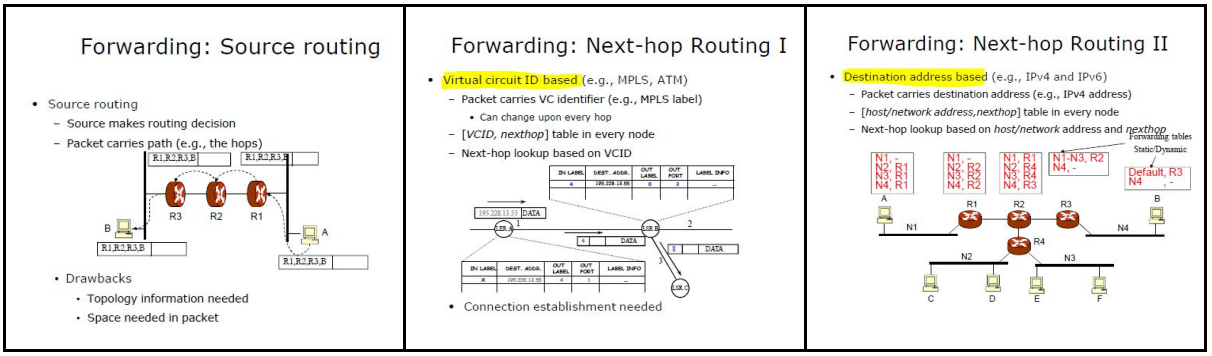
Connectionless (只规定destination, 不规定具体路线): Packets may arrive in different paths, and may not receive in order.

IP: Implemented using unicast addresses shared between several hosts.

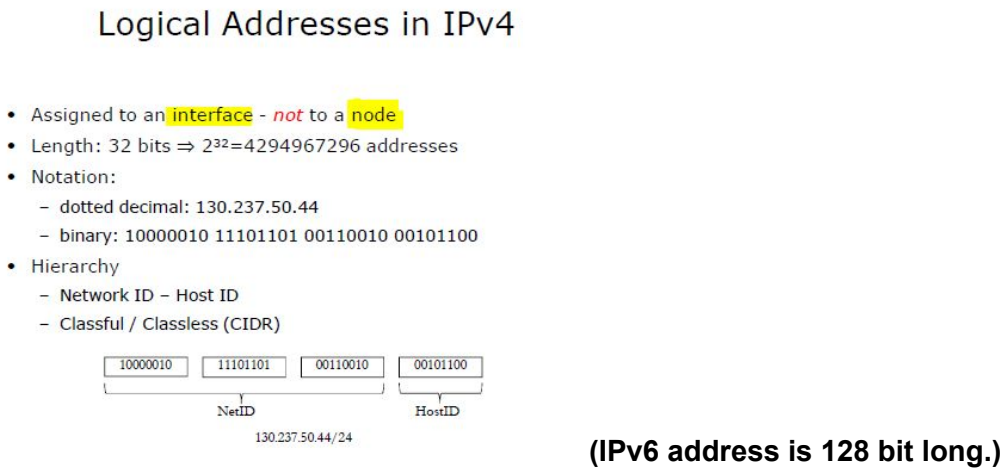
Direct Dilivery	Indirect Dilivery
 **Host to host** or **router to host**	 **Host to router** or **router to router**
– Destination and sender connected to the same physical network • Last delivery is direct – Destination address and local interface have same network address (use netmask)	– From host to router or from router to router – Destination address and forwarding table: forwarding

2. Forwarding

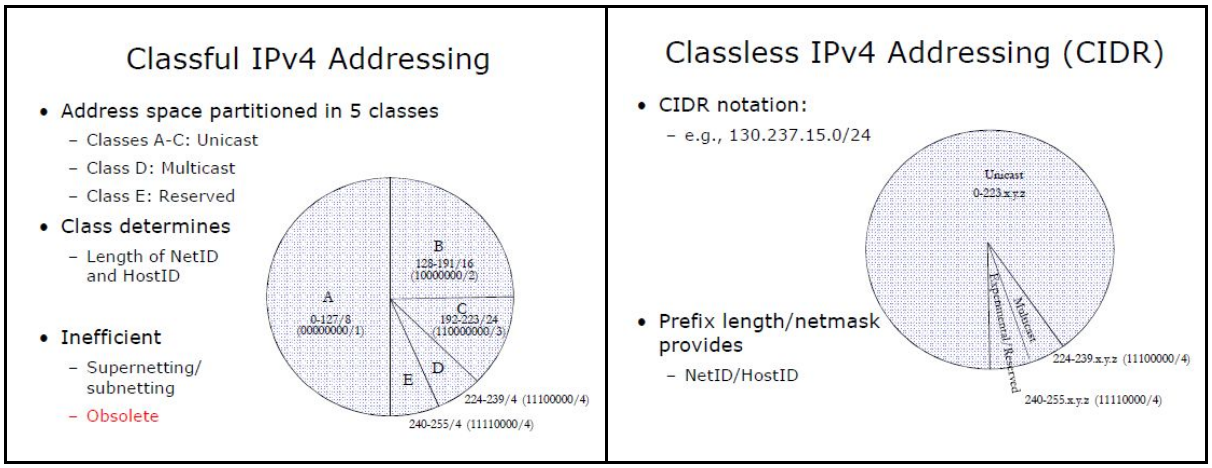
The last one is using nowadays.



3. IPv4 Addressing

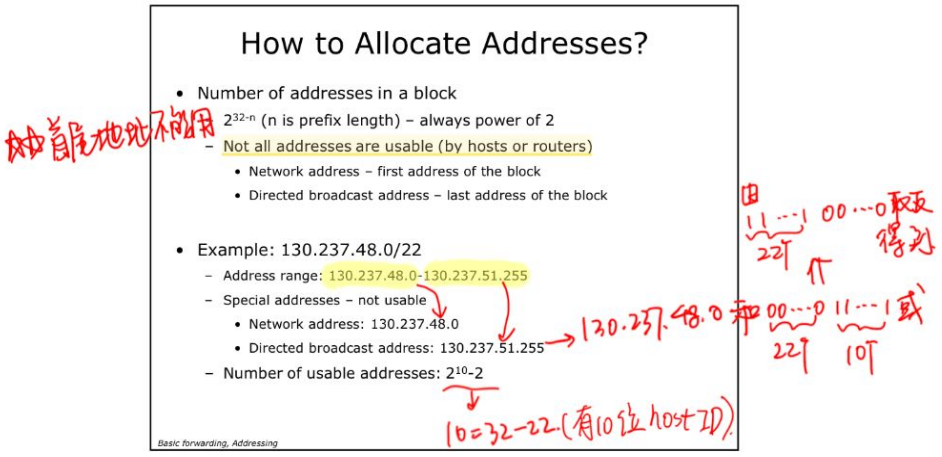


(IPv6 address is 128 bit long.)

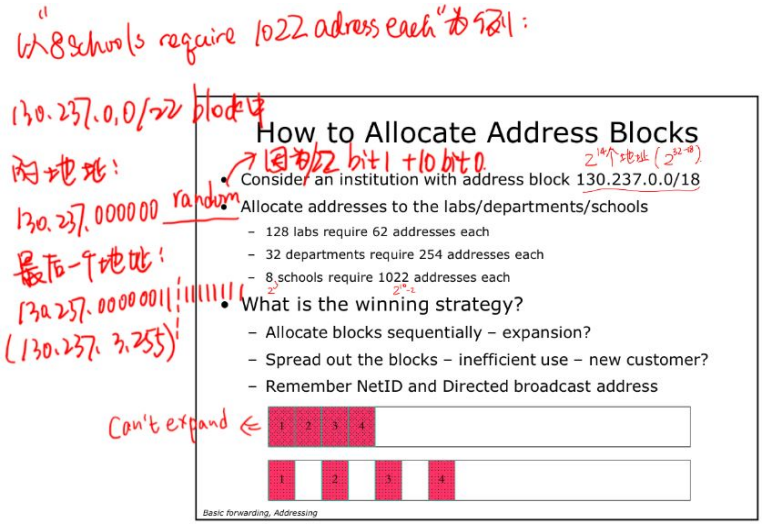


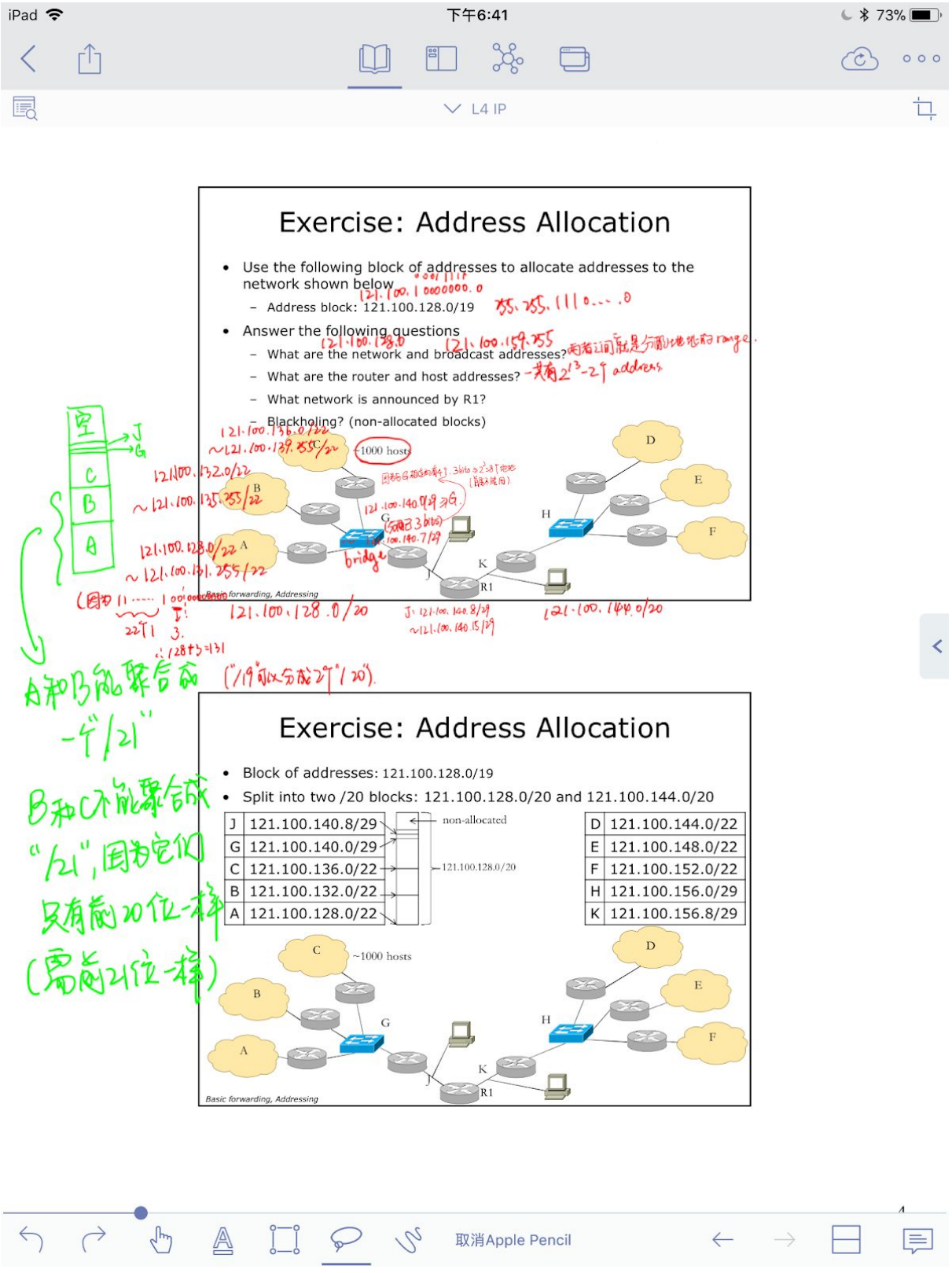
– Address & Mask = NetID (network address)
– Address & !Mask = HostID (host address)
– Address | !Mask = Directed broadcast address
（ limited broadcast address是都set to 1，255.255.255.255 ）

4. Allocate address



Address range 是从 **network address** 到 **broadcast address** 的.





Reading instruction:
Ch 5,12.2,26

Lecture 4 IP

1. IPv4&IPv6 packet

Needed for packet processing

IPv4

- Header Length (4 bits)
  - Size of IPv4 header including options (20-60 bytes)
  - Granularity: 4 bytes
    - 5≤HLEN≤15
    - limits header size (20 ≤HS ≤60)!!!
- Total Length (16 bits)
  - Total length of datagram including header (20-65535 bytes, practice ≤ 8KB)
  - If datagram is fragmented: length of fragment
  - Granularity: 1 byte

IPv6

- Payload Length (16 bits)
  - Total length of payload excluding base header (0-65535 bytes)
  - If datagram is fragmented: length of fragment
  - Granularity: 1 byte
  - Extendable: Jumbo payload up to 4GB (see later)

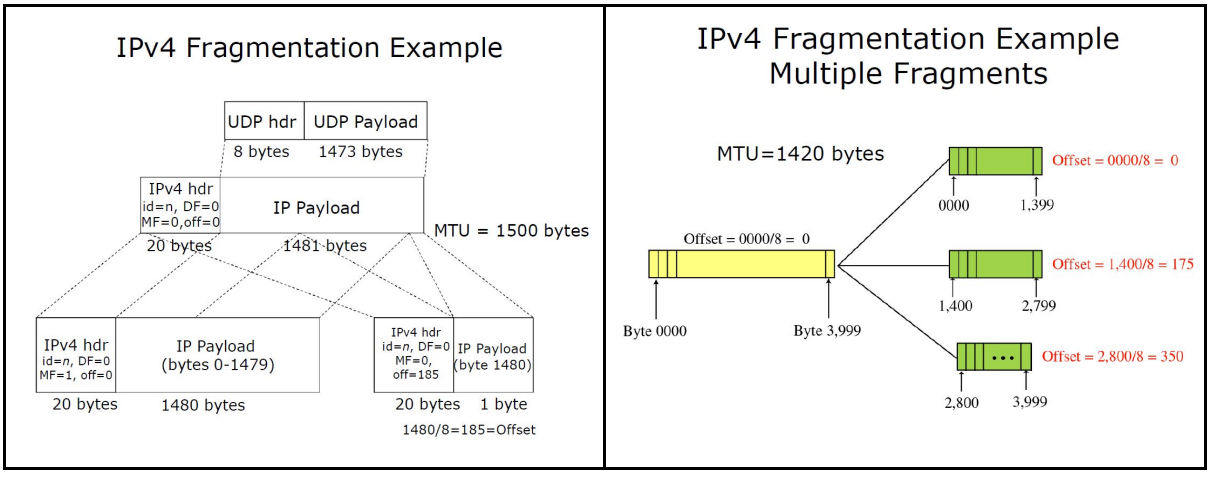
Packet contains source and destination addresses.

IPv6 没有 header length field (因为长度(不包括extension header)是固定的).

2. IPv4&IPv6 Fragmentation

MTU是数据链路层的限制，但是fragmentation发生在网络层

注意**UDPheader只有8 bytes长**



Flags:

– RF (Reserved Fragment) – for future use (set to 0)

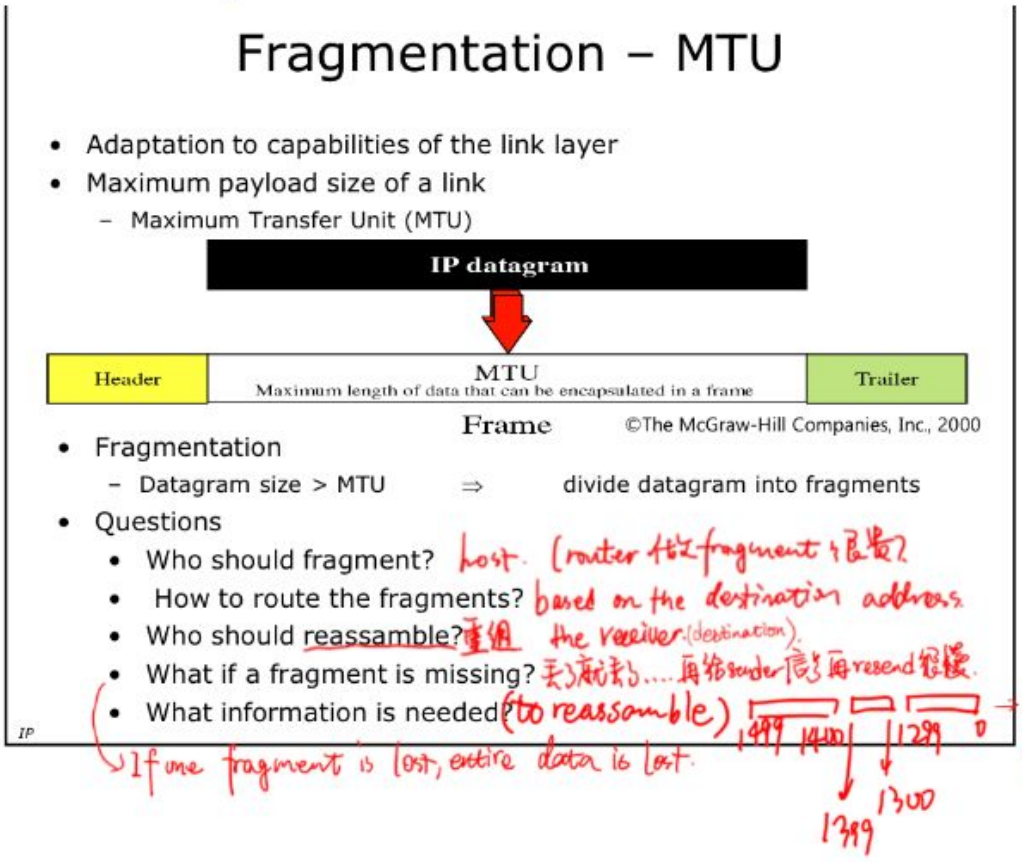
– DF (Dont Fragment).

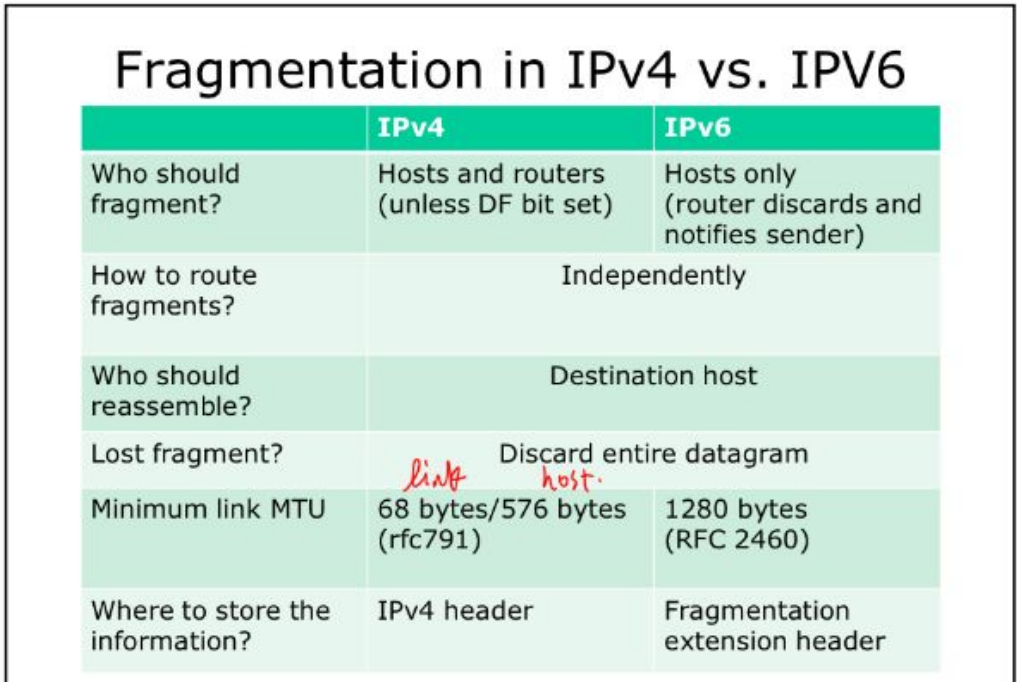
• Set to 1 if datagram should not be fragmented.

• If set and fragmentation needed, datagram will be discarded and an error message will be returned to the sender

– MF (More Fragments)

• Set to 1 for all fragments, except the last.





IPv6只能在source，不能再中间的router上面做fragmentation

3. IPv4 header vs IPv6 header

	IPv4	IPv6

header length	20-60 bytes	40 bytes
header length field	有	没有 (header length fixed)
name of down-counter lifetime of every datagram (可以自行设置, Default initial value: 64)	**TTL** (8 bit) 每经过router, -1. =0时丢弃 Every router holding a datagram for more than 1 second should decrement the TTL by the number of seconds	**Hop Limit** (8 bit)
	connectionless	**connection-oriented** Because of the adding of **flow label** (textbook P830)

Comparison between IPv4 and IPv6 Headers

The following shows the comparison between IPv4 and IPv6 headers.

- The header length field is eliminated in IPv6 because the length of the header is fixed in this version.
- The service type field is eliminated in IPv6. The traffic class and flow label fields together take over the function of the service type field.
- The total length field is eliminated in IPv6 and replaced by the payload length field.
- The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
- The TTL field is called hop limit in IPv6.
- The protocol field is replaced by the next header field.
- The header checksum is eliminated because the checksum is provided by upper layer protocols; it is therefore not needed at this level.
- The option fields in IPv4 are implemented as extension headers in IPv6.

IP Network layer functions	**IPv4实现方法**	**IPv6实现方法**
Logical addressing	Locating hosts	
Routing	Path determination	
Forwarding	Move Packet from input to output of the routers	
Fragmentation	Adaption to lower layer	
Multiplexing/demultiplexing （多路复用和解复用）	Many transport layer protocols	

Error detection + avoidance	checksum	-
QoS (Quality of Service)	ToS (Type of Service): 8 bit	[1]Traffic Class: 8 bit Flow Label: 20 bit

Reading instruction:
Ch 6,7,27

Lecture 5 IP, ARP, more IP and ICMP

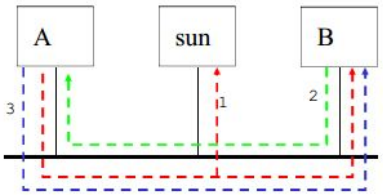
1. ARP[2]

已知IP address，想得到MAC address：ARP（Address Resolution Protocol）
已知Name（域名，比如kth.se)，想得到IP address： DNS

ARP Example (IPv4)

A wants to send an IP datagram to *B* (140.252.13.34)

1. Send an *ARP request* on broadcast to all stations:
   - "Who has 140.252.13.34?"
2. *B* identifies it as its own address and sends an *ARP reply* on unicast back to *A*
   - "I have 140.252.13.34 and my mac address is 0:0:c0:c2:9b:26"
3. *A* sends the datagram to *B* using the resolved MAC address
4. Note that *sun* and *B* can update their ARP caches with *A*!



注意这个过程在高速缓存cache中存储了，所以第二次传输datagram的时候不需要再进行APR request (broadcast) 和APR reply (unicast)。

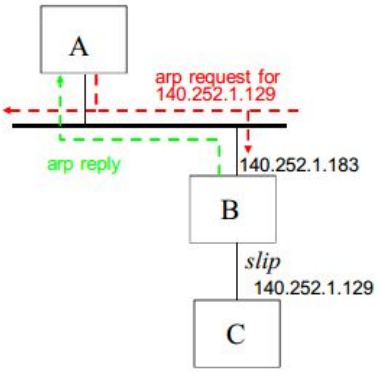
ARP Packet

- Two length fields
  - Hardware (Ethernet address length: 6) Ethernet address的长度是48bit
  - Protocol (IP address length: 4) IP address的长度是32bit
- Sender hardware (e.g., *Ethernet)* and protocol (e.g., *IP*) address
- Target hardware (e.g., *Ethernet)* and protocol (e.g., *IP*) address
- ARP packet encapsulated into a link layer frame (e.g., Ethernet)

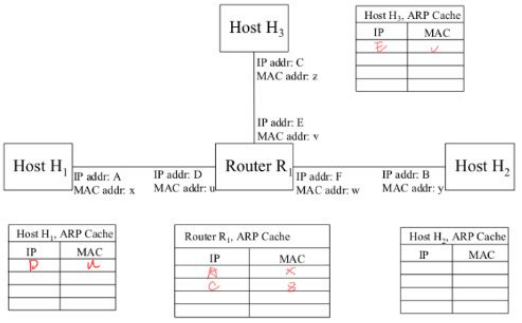
[1] The flow label field allows labeling packets belonging to a particular flow for which the sender requests the same handling. A flow is identified by the source address, destination address, and a nonzero flow label, and packets belonging to a flow should be treated the same way by a router (rfc6437)
The first 6 bits of the traffic class field hold the DiffServ Code Points (DSCP), which can be used for classifying packets in order to implement priorities. The last 2 bits are used for ECN, and serve for signaling congestion in the network. (rfc3260)
[2] Address Resolution的方式包括：
(用于IPv4的Address Resolution Protocol – ARP，这一部分中介绍)
(用于IPv6的Neighbor Discovery Protocol – ICMPv6，单独在3. ICMP中介绍)

代理ARP：

- Proxy ARP
  - Respond to ARP requests on someone else's behalf
- Allows sub-networks to be hidden
- Example
  - *C* is hidden behind *B*: *B* responds on behalf of *C*



ARP - Example



- Three hosts H1, H2 and H3
- Routed network running IPv4
- The IP and MAC addresses of the hosts and the router's interfaces are given in the figure.
- The ARP caches of the hosts and the router are shown. Assume that the ARP caches are initially empty and that no packets have been sent yet.

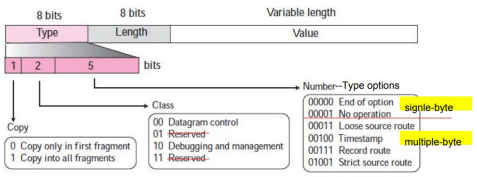
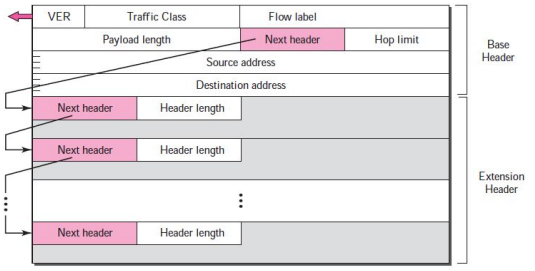
- Host H1 wants to send an IPv4 unicast datagram to host H3.
  - Fill in the state of the four ARP caches as they will appear after the IPv4 unicast datagram has been delivered to host H3, that is, after dynamic ARP resolution has been made.

2. IP options (extension header)

	IPv4	**IPv6**
Purpose	Control, testing and debugging of the network functionality	
Length	IPv4: Max 40 bytes[3] (Max header length is 60 bytes in IPv4)	IPv6: No limitations[4]

[3] The header of the IP datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises（包括） the options, which can be a maximum of 40 bytes.

[4] The length of the base header is fixed at 40 bytes. However, to give more functionality to the IP datagram, the base header can be followed by up to six extension headers.

<table>
<tr><td>Option Format</td><td> 
</td><td> 
</td></tr>
<tr><td rowspan="6">Option Type</td><td>End of option</td><td>Hop-by-hop options
(Pad1&PadN&jumbao payload)
用于当source需要把信息传递给datagram经过的所有routers时. eg,.if the length of the datagram is more than the usual 65535 bytes.
❑ Pad1 这个选项是 1 字节长，其作用是为了对齐。某些选项需要在 32 位字的特定位开始（见后面的特大有效载荷的描述）。如果选项正好差一个字节才能满足这个需求，则可加上 Pad1 来填补这个差额。Pad1 既不包含选项长度字段，也不包含选项数据字段。它仅包含了一个把所有位都置为 0 的选项代码字段（动作是 00，改变位是 0，类型是 00000）。Pad1 可在逐跳选项首部中的任何地方插入（见图 27.7）。
❑ PadN PadN 在概念上与 Pad1 相似。不同的地方是当两个或更多字节需要对齐时就要使用 PadN。这个选项包括 1 字节的选项代码、1 字节的选项长度和可变数目的零填充字节。这个选项的代码值是 1（动作是 00，改变位是 0，类型是 00001）。选项长度包含的是填充字节的数目，见图 27.8。
❑ 特大有效载荷（jumbao payload） 我们已经讲过，IP 数据报的最大有效载荷长度是 65 535 字节。但是，如果由于某种原因需要使用更长的有效载荷，我们就可以使用特大有效载荷选项来定义这个更大的长度。特大有效载荷选项必须总是从扩展首部的最前端算起的 4 字节的倍数再加上两个字节的地方开始。也就是说特大有效载荷选项在 (4n + 2) 字节处开始，这里的 n 是一个小整数，参见图 27.9。</td></tr>
<tr><td>No operation</td><td>Routing</td></tr>
<tr><td>Loose source route[5]</td><td>Fragment</td></tr>
<tr><td>Timestamp</td><td>Authentication Header</td></tr>
<tr><td>Record route</td><td>Encapsulating Security Payload</td></tr>
<tr><td>Strict source route</td><td>Destination options</td></tr>
</table>

[5] 对其中的Source Route进行说明：预先指定数据报在因特网中传送时的路由，分为strick和loose两种
– Strict Source routing (SSRR)
• The path is exactly as specified - 只能访问sender规定的路由。访问其他路由时，该路由会丢弃datagram然后发送error message
– Loose Source Routing (LSRR)
• The path includes the specified addresses - 必须访问sender规定的路由，其他路由也可以访问
Source route: when it comes handy
• Troubleshooting
– Figure out from point "A" why machines “B" and “C" cannot communicate
• Mapping the network
– Used with traceroute in order to find all the routes between two points on the network
• Performance
– Force an alternate link to avoid congesting the correct routes w/o changing the forwarding tables (management)
– Create independent paths for MDC or FEC
• Hacking
– Can send packets to a host via a trusted third party
• Normally disabled in routers…

对比

IPv4 options *vs.* IPv6 extension headers

IP diagnosis and control

Purpose	IPv4 Option	IPv6 Extension header
Source routing	Loose/Strict Source Route and Record	Routing
Route recording	Record Route	-
Fragmentation	-	Fragment
Jumbograms		Jumbo payload option (in Hop-by-hop options)
Delay measurement	Timestamp	-
Special attention	Router alert *(proposed)*	Router alert option (in Hop-by-hop options)
Security	AH, ESP* (as payload)	AH, ESP

（IPv4的fragmentation在fixed header里面实现）

3. ICMP

（不属于TCP也不属于UDP）

ICMP Summary

Error reporting	ICMPv4	ICMPv6
Destination unreachable	Yes	Yes
Source quench	Yes	No
Packet too big	No	Yes
Time exceeded	Yes	Yes
Parameter problem	Yes	Yes
Redirect	Yes	Yes

Query/Informational/NDP/MLDP	ICMPv4	ICMPv6
Echo request and reply	Yes	Yes
Timestamp request and reply	Yes	No
Address mask request and reply	Yes	No
Router solicitation and advertisement	Yes	Yes
Neighbor solicitation and advertisement	ARP	Yes
Group membership management	IGMP	Yes

ARP,IP,ICMP

Tool using ICMP: traceroute
Explore every IP hop on the way.

ICMP {ICMPv4 / ICMPv6}
(Report IP problems back to sender)
Control and Management.

Tool using ICMP: Ping
Test host reachability

2 types.

Error Reporting Message
(只report，不做 correction.)
(更高层 protocol 做 correction)

Query / Informational messages.

作用：
① send back to source (用到 IP datagram 中的 source address)
② carries part of original datagram header and payload.

③ 不能用于：

ICMP Error not returned for
- A datagram carrying another ICMP Error
- A datagram destined to IP broadcast or multicast address
- A datagram sent as link-layer broadcast (e.g., Ethernet)
- An IP fragment other than the first
- A datagram whose source address does not define a single host (e.g., 0.0.0.0)

Reason
- Avoid loops
- Avoid packet explosions (broadcast storms)

其中2种：

① ICMP Time Exceeded (both IPv4 & IPv6)

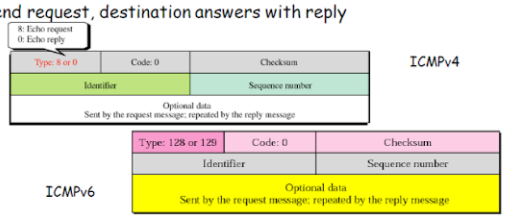
Sent in 2 cases
- Router
  - When TTL is zero after decrementation, discards the datagram and sends this back to the source (Code 0)
- Destination host
  - When all fragments of a datagram do not arrive at the destination host within a certain time limit (Code 1)
    - Timer is started upon reception of the first fragment

② Packet too big (only IPv6).
i. datagram 比 MTU 大 { discard
send message (inc. MTU) back to source
ii. can be used for path MTU discovery

类似 IPv4 Destination unreachable (fragmentation needed) → UDP packet

其中2种：

① ICMP Echo Request and Reply (both IPv6 & IPv4)
- Purpose: check host reachibility
- Operation
  - Send request, destination answers with reply

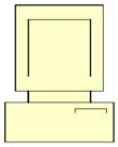


② Neighbor Discovery Protocol (only IPv6)
- Purpose
  - Assist forwarding of datagrams
    - Address resolution (ARP in IPv4)
    - Forwarding table updates (Redirect in ICMPv4)
    - Router address
- Operation
  - Neighbor solicitation = query
    - Includes the solicitor's physical address in the solicitation for easier processing at the receiver
  - Neighbor advertisement = reply
  - Redirect = forwarding table update necessary
  - Router solicitation/advertisement

Reading instruction:
Ch 7-9, 27, 28

Lecture 6&7 Routing

1.In general



- Host (end-system)
  - One or many network interfaces
  - Can *not* forward packets between interfaces

- Router
  - Two or more interfaces
  - Can forward packets between interfaces
  - Forwards on Layer 3 (IP Layer)

•Problem

–Find best path from router to host

•Typically based on shortest path algorithms (from graph theory)

• Bellman-Ford algorithm

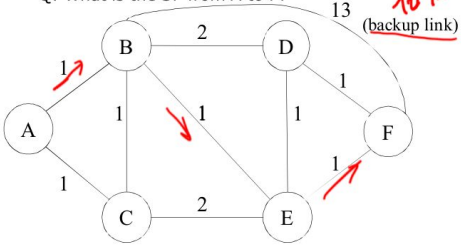
– Used by Distance-Vector protocols (RIP, IGRP, BGP)

• Dijkstra's algorithm

–Used by Link-State protocols (OSPF,popular among organizations, IS-IS,p.a. operators)

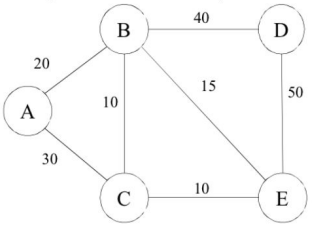
Shortest Path (SP) Problem

- Given weighted graph *G(V,E)* and nodes (*s,d*)
  - *Weight denotes cost*
- Find an (*s,d*) path with minimal path cost
- Equal cost multipath (ECMP)
  - Set of paths with the minimal cost
- Q: What is the SP from A to F?



Alternative: Widest Path Problem

- Numbers denote *width*
  - *e.g., available* bandwidth
- Find path with maximum width
  - Also called "Unsplittable maximum flow" problem
- SP algorithms can be modified to solve widest path problem
- Q: What is the widest path from A -> E?



2. Bellman-Ford Algorithm

Distributed Bellman-Ford Algorithm and DV protocols

Distant Vector

- Data structure at node *r*: *"distance vector" table*
  - One entry for every destination *d* in the network
  - For every *d* stores the metric *M(r,d)* (distance) and the next-hop *n∈N(r)*

Dest	Cost	NextHop
...	...	...

- Periodic message exchange
  - Send (Dest,Cost) of the table (distance vector) to all neighbors
- For each update that comes in from a neighbor *n'∈N(r)* (with a metric *M(n',d)* to destination *d*)
  1. Compute *m=M(r,n')+M(n',d)*
  2. if (*m<M(r,d)) then n=n', M(r,d)=m*
  3. *elseif (n=n') then M(r,d)=m %new value from same*
- In DV protocols M is bounded, typically to 16
  - The upper bound is defined as unreachable (infinity)

Distance-vector = (destination, metric, next-hop) [metric表示的是cost的度量]
Path-vector = (destination, path, next-hop)

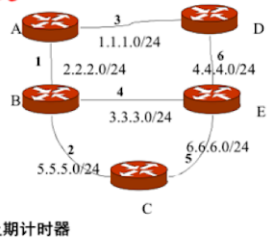
3. Real network中的问题和解决

RIP: Routing Information Protocol.

Going to real networks

- IP networks require destinations and nexthops (not just nodes)
  - Destinations are networks (e.g., 192.16.32.0/24)
  - Next-hops are IP addresses (e.g., 192.16.32.1)
- Suppose the topology changes, e.g., routers, links crash?
  - Use timers (counters) and age the entries
  - If you do not hear from a router in (e.g.) 180s, mark it as invalid
  - Send updates every (e.g.) 30s

计时器 中文书P286



Dest	Cost	NextHop
1.1.1.0/24	3	-
2.2.2.0/24	1	-
3.3.3.0/24	5 (B)	2.2.2.2
4.4.4.0/24	9 (D)	1.1.1.2
5.5.5.0/24	3 (B)	2.2.2.2
6.6.6.0/24	8 (B)	2.2.2.2

Converged routing state of A

21

截止期计时器

截止期计时器（expiration timer）管理路由的有效性。当路由器收到路由更新信息时，截止期计时器就为这个特定的路由设置到 180 秒。每当收到该路由的一个新的更新时，截止期计时器就要复位。在正常情况下，每隔 30 秒发生一次复位。但是，如果互联网中出了问题，并且在分配的 180 秒内没有收到任何更新报文，那么这个路由就被认为是过期了，而路由的跳数就被设置为 16，这表示终点不可达。每一条路由有它自己的截止期。

Distance Vector Problem: Count to Infinity (Two-node instability)

定义无穷大. 中文书P281

路径变短而"坏消息传得快"

但路径变长而"好消息传得慢"

容易造成循环

习惯置无穷大(16)

Initially, R_1 and R_2 both have a route to N with metric 1 and 2, respectively.

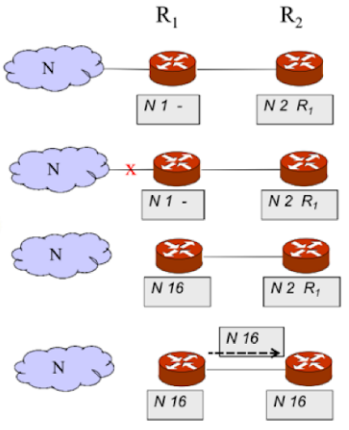
The link between R_1 and N fails.

R_1 removes its route to N, by setting its metric to 16 (infinity).

即网络的大小不能超过15hop.

One of two things can happen:

1) R_1 reports its route to R_2. Everything is fine.



22

10

Distance Vector Problem: Count to Infinity (Two-node instability)

2) R_2, which still has a route to N, advertises it to R_1.

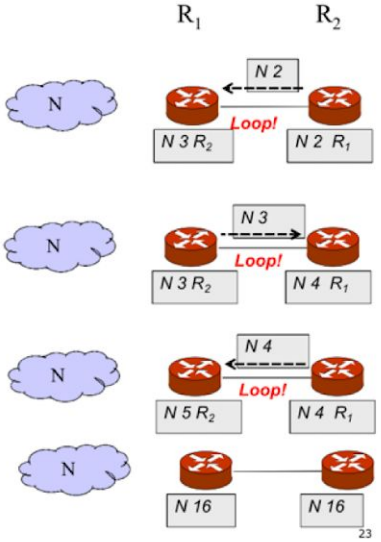
Things start to go wrong: packets to N loop until their TTL expires!

Eventually (~10-20s), R_1 sends an update to R_2. The cost to N increases, but the loop remains.

Yet some time later, R_2 sends an update to R_1.

...

Finally, the cost reaches infinity at 16, and N is unreachable. The loop is broken!



23

法1. 分割范围.

(洪泛). 不flooding, 发送整个routing table 表的最性路径如果经过R1, R2将不会把它一行发给R1.

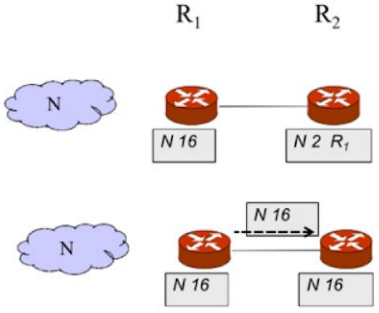
Solution: Split Horizon

- *Do not send routes back over the same interface from where the route 'arrived'.*
  - –Helps to avoid "mutual deception": two routers tell each other they can reach a destination via each other.

这是产生上图Loop的根源.

R_2, does not announce the route to N to R_1 since that is where it was learnt from.

Eventually, R_1 reports its route to R_2 and everything is fine.



24

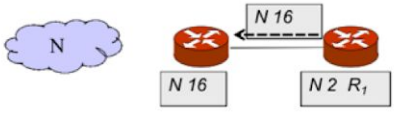
法2. 分割范围+毒性反转.

Solution: Split Horizon + Poison Reverse

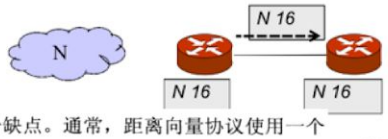
- *Advertise reverse routes with a metric of 16 (i.e., unreachable)*
  - Does not add information but breaks loops faster
  - Adds protocol overhead

R_1 R_2

R_2 always announces an unreachable route for N to R_1.



Eventually, R_1 reports its route to R_2 and everything is fine.



25

分割范围和毒性逆转 使用分割范围策略有一个缺点。通常，距离向量协议使用一个计时器，若长时间没有关于某个路由的消息，就要从路由表中删除这个路由。在前面描述的场景中，当结点 B 在它给 A 的通告中删除了到 X 的路由时，结点 A 并不能猜出这是由于分割范围策略（因为信息的来源是 A），还是因为 B 最近一直都没有收到有关 X 的任何消息。分割范围策略可以与**毒性逆转**（poison reverse）策略组合起来使用。（译者注：这里的“毒性”表示度量为 16，表示网络中出现了故障。）结点 B 可以仍然通知关于 X 的数值，但如果信息源是 A，就把距离换成为无穷大作为一种警告：“不要使用这个数值，我所知道的关于这条路由的信息来自于你。”

Remaining problems

- More than two routers involved in mutual deception
  - –A may believe it has a route through B, B through C, and C through A
- Split horizon with poison reverse does not help ☹

三节点的不稳定性.

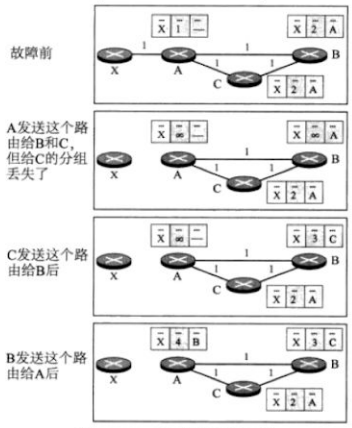


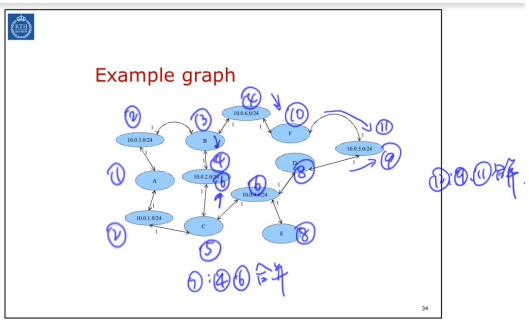
图 11.9 三结点的不稳定性

假定在发现 X 不可达之后，结点 A 发送分组把这个情况通知 B 和 C。结点 B 立即更新其路由表，但发送给 C 的分组在网络中丢失了，因而永远不能到达 C。结点 C 完全不了解情况，仍然以为有一条距离为 2 的路由能够经过 A 到达 X。过了一会，结点 C 把它含有到 X 的路由的路由表发送给 B。此时结点 B 完全被愚弄了，它收到的信息是说有一条路由可以经过 C 到达 X。结点 B 根据算法更新了自己的路由表，指出有一条代价为 3 的路由可以经过 C 到达 X。这个信息来自 C 而不是 A，因此过一会结点 B 可能会把这个路由通知 A。现在连 A 也被愚弄了，它更新了自己的路由表，指出从 A 出发可以经过 B 到达 X，代价是 4。当然，这个循环会继续下去。现在 A 又把到 X 的路由的代价增加后通知给 C，但是不会给 B。C 又把增加了代价的路由通知给 B。B 向 A 又做了相同的事，如此循环往复。当每一个结点的代价都达到无穷大时，这个循环就结束了。

26

4. Dijkstra's shortest path algorithm

例子：



Permanent set	Tentative set
A 0 -	10.0.3.0/24 1 - 10.0.1.0/24 1 -

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 -	~~10.0.3.0/24 1 -~~ 10.0.1.0/24 1 - B 1 -

choose the least cost to permenant set

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 -	10.0.1.0/24 1 - ~~B 1 -~~ 10.0.2.0/24 2 B 10.0.6.0/24 2 B

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 -	~~10.0.1.0/24 1 -~~ 10.0.2.0/24 2 B 10.0.6.0/24 2 B C 1 -

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 -	10.0.2.0/24 2 B 10.0.6.0/24 2 B ~~C 1 -~~ 10.0.2.0/24 2 C 10.0.4.0/24 2 C

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 - 10.0.2.0/24 2 B 10.0.2.0/24 2 C	~~10.0.2.0/24 2 B~~ 10.0.6.0/24 2 B ~~10.0.2.0/24 2 C~~ 10.0.4.0/24 2 C

Note: ECMP

ECMP: Equal Cost MultiPath. More than one path.

(指一个Network只是来自不同 neighbor.

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 - 10.0.2.0/24 2 B 10.0.2.0/24 2 C 10.0.4.0/24 2 C	10.0.6.0/24 2 B ~~10.0.4.0/24 2 C~~ D 2 C E 2 C

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 - 10.0.2.0/24 2 B 10.0.2.0/24 2 C 10.0.4.0/24 2 C E 2 C D 2 C	10.0.6.0/24 2 B D 2 C E 2 C 10.0.5.0/24 3 C

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 - 10.0.2.0/24 2 B 10.0.2.0/24 2 C 10.0.4.0/24 2 C E 2 C D 2 C 10.0.6.0/24 2 B	10.0.6.0/24 2 B 10.0.5.0/24 3 C F 2 B

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 - 10.0.2.0/24 2 B 10.0.2.0/24 2 C 10.0.4.0/24 2 C E 2 C D 2 C 10.0.6.0/24 2 B F 2 B	10.0.5.0/24 3 C F 2 B 10.0.5.0/24 3 B

Permanent set	Tentative set
A 0 - 10.0.3.0/24 1 - B 1 - 10.0.1.0/24 1 - C 1 - 10.0.2.0/24 2 B 10.0.2.0/24 2 C 10.0.4.0/24 2 C E 2 C D 2 C 10.0.6.0/24 2 B F 2 B 10.0.5.0/24 3 B 10.0.5.0/24 3 C	10.0.5.0/24 3 C 10.0.5.0/24 3 B

Note: ECMP

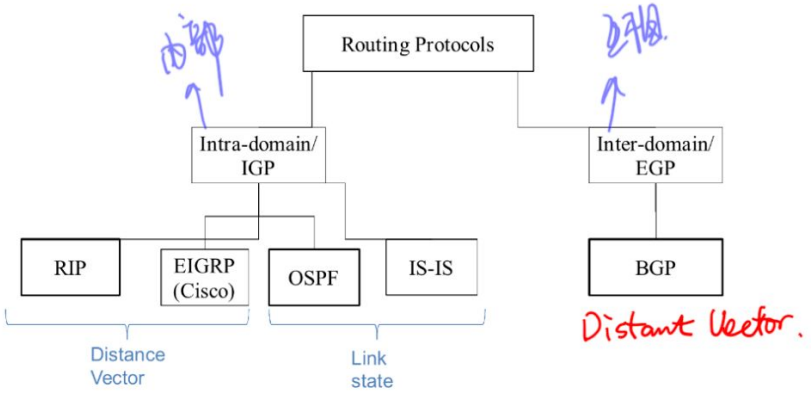
5. Link-state vs. Distance-vector

Link-state vs. Distance-vector

- Distributed database model
- Advantages
  - –More functionality due to distribution of original data
  - –No dependency on intermediate routers
  - –Easier to troubleshoot
  - –Fast convergence: when the network changes, new routes are computed quickly
  - –Less bandwidth consuming

- Distributed processing model
- Advantages
  - –Less complex – easier to implement and to administer
  - –Needs less memory

6. Routing Protocols



1) RIP

•Metric: hop count (1: directly connected, 16: infinity)
•Supports networks with diameter 小于等于15
•Timeout timer (Purge清除 routes that are not refreshed)
•Messages carried in UDP datagrams
–Broadcast (RIP-1)
–IP Multicast (RIP-2): 224.0.0.9
–IPv6 Multicast (RIPng): FF02::9

2) Open Shortest Path First protocol (OSPF) 相对RIP更适合用于大一点的网络

与RIP和BGP不同的是，OSPF协议不使用TCP或者UDP协议而是承载在IP协议之上

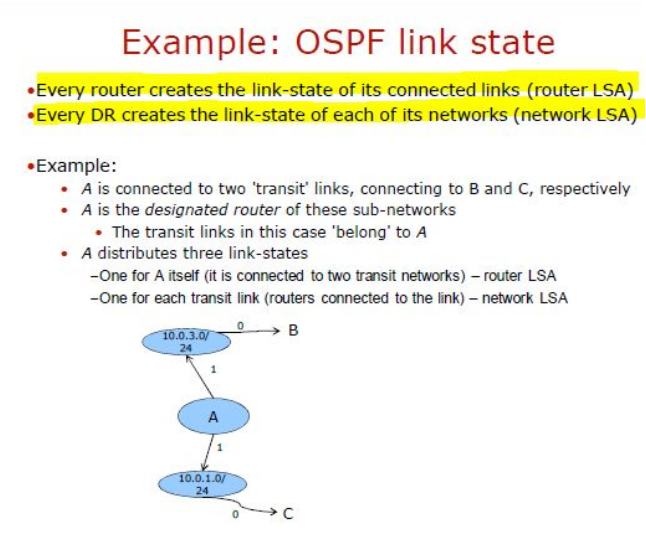
• Metric: arbitrary (Often related to link speed (inverse proportional))
• Scaling achieved through hierarchy
- Every network segment has 1 designated router (+1 backup) – DR, BDR (每条线上1DR)
- AS split into areas – use Dijkstra for an area
• Messages carried directly on top of IP
- IP Multicast: 224.0.0.5
- IPv6 Multicast: FF02::5
• OSPF protocol components

(1) Hello protocol
- Detection of neighboring routers
- Election of designated router (and backup) adjacency

(2) Exchange protocol
- Exchange link-state between adjacent routers

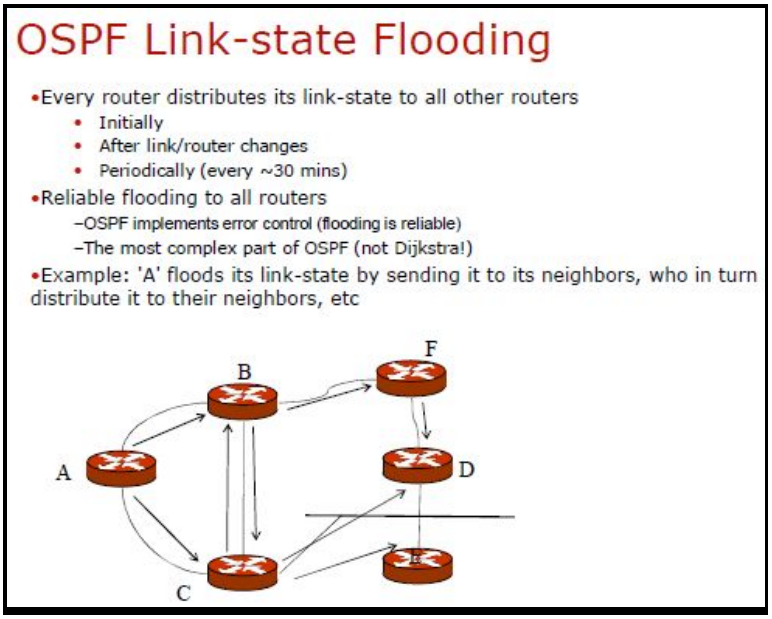
(3) Reliable flooding
- When links change/age: send update to adjacent routers and flood recursively

(4) Shortest path calculation
- Compute shortest path tree to all destinations using Dijkstra's algorithm



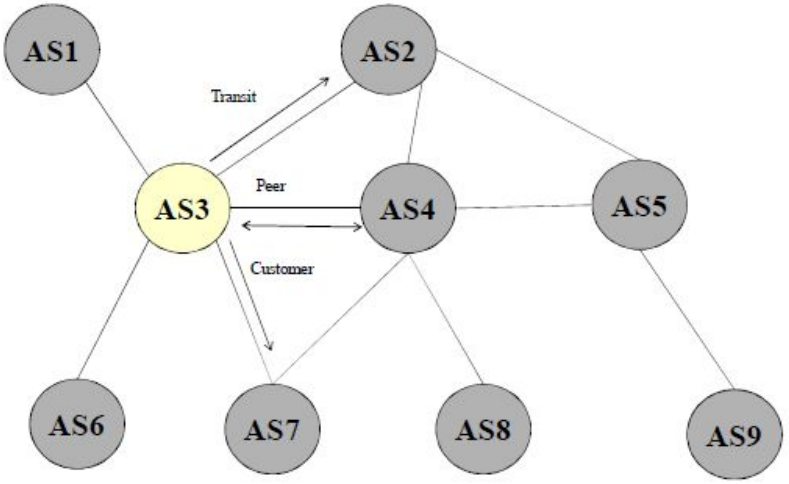
important example[6]

[6] LSA: link state advertisement



3) Inter - domain

AS（Aotonomous Systems）的概念： A set of routers. 每个AS都有AS number



AS7是AS3的customer，have to pay to AS3

4) Border Gateway Protocol (BGP) v4

•Path-vector routing protocol (和之前的两个Link-state, Distance-vector都不一样)

•Path vector consists of AS:s, not IP addresses

–Hides internal structure in the domains

–Loop detection only on AS-numbers!

–Example: <dst: 10.1.10/24, path: AS1:AS3:AS5, nexthop: 10.2.3.4>

•Used between domains (AS:s)

• Views the Internet as a collection of AS:s

•Supports the destination-based forwarding paradigm

–Other relations are not expressed: sources, tos, link load

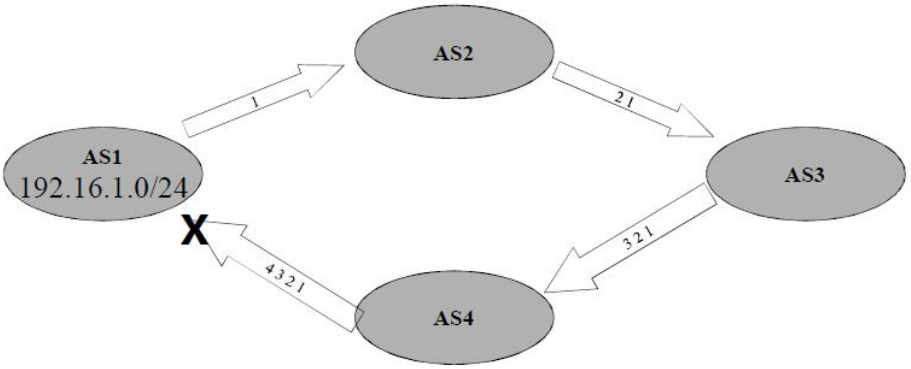
•Uses TCP for data transmission between BGP peers

•Tags destinations with path attributes （attribute，属性，定语）

•Describe different properties of the destination (e.g., preferences)

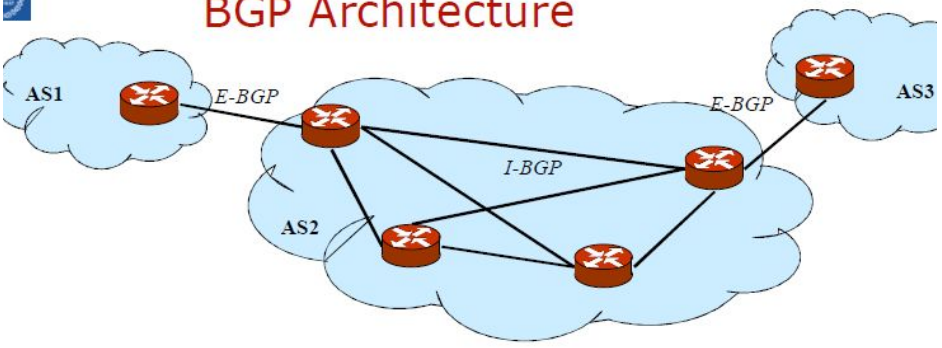
•Can express and enforce policy decisions at AS level

AS-PATH attribute



- AS-PATH used to break loops (between AS:s)
- AS1 announces 192.16.1.0/24 to AS2 and detects its own ASN when received from AS4
- AS-PATH is the most well-known path-attribute, there are several others

BGP Architecture



- BGP has two uses/variants
  - E-BGP: exchanges external routes between border routers *between* AS:s
  - I-BGP : synchronises *external* routes *within* an AS (IGP takes care of internal routes)
- BGP interacts with Internal routing (OSPF/IS-IS/RIP/...)
  - *Redistributes* internal / external routes between the two protocols

Reading instruction:

Chapter 11: Unicast routing protocols

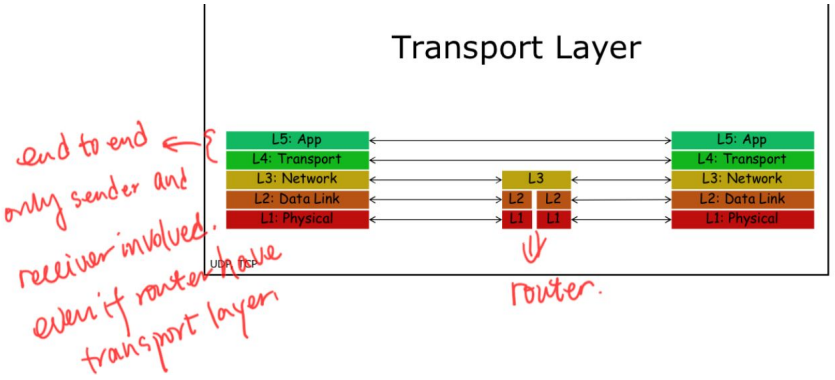
You need to complement with slides, especially if you do not make the routing lab

11.6 OSPF: Skip detailed packet descriptions

11.8 BGP: Skip detailed packet descriptions

Lecture 8&9 Transport Layer - UDP, TCP and beyond

1.In general



Purpose： Logical process-to-process communication
和IP Layer的对比：

IP	UDP	TCP
connectionless （只规定destination）	connectionless	connection - oriented （规定具体路线）
unreliable	unreliable	reliable
message- oriented （Deliver the whole packet together, 传或不传）	message- oriented	byte - stream oriented （the unit of transmission is byte, and in order）
		full-duplex

2. UDP

▵UDP data delivery based on: receiver port number and receiver IP address.
▵UDP checksum: IPv4可以有，IPv6必须有（IPv6在IP层没有checksum）
▵psedo-header：to double check
▵Maximum UDP Segment Size
• Theoretical limit
– IPv4: 65,507 bytes of payload
• 65,535 bytes (IPv4) - 20 bytes IP header - 8 bytes of UDP header
– IPv6: 65,527 bytes of payload
• 65535 bytes payload – 8 bytes UDP header

3. TCP

▵TCP service model
1) Connection-oriented – a virtual circuit[7]
2) Between exactly two end-points – Broadcast and multicast不能用TCP (use UDP)
3) Full duplex

[7] 虚拟电路（英语：Virtual circuit，缩写为 VC），又称为虚电路、虚连接或虚通道，在分组交换的电脑网路上，交换资料的传输方式之一。它是一种预接式（connection-oriented），或线路交换式（circuit-switched）的资料传输方法，在两个终端系统（End system）间，建立一条连线，来进行资料交换。

4) Reliable and in-order – Delivery is not guaranteed but reception is known
5) Byte stream service
– A stream of 8-bit bytes is transmitted over the TCP connection
– No record markers inserted by TCP
△TCP service function
1) Multiplexing/Demultiplexing[8]
• TCP connection identification (UDP不需要sender的IP address和port number)
– Sender IP address and port number
– Receiver IP address and port number
2) Segmentation - Byte stream to segment translation[9]
Try to send as big segments as possible (MSS) ：
• The largest chunk of data TCP will send to the other side
– Can be announced in the options field of the TCP header during connection establishment
• If not announced, a default value is assumed
– 576 bytes host MTU requirement in IPv4 : 536 bytes
– 1280 bytes MTU requirement in IPV6: 1220 bytes
• Large MSS means
– Less overhead[10] (headers)
– Less segments to take care of (will see later)
• Until fragmentation occurs (Path MTU discovery)
– Potentially more delay
3) Error control - Reliable transmission over unreliable channel
– Noise → Bit error → Packet corruption → Packet drop
– Congestion → Packet drop
三种flavor：
①Stop-and-wait[11]
Both the sender and the receiver use a sliding window of size 1. 只有两个序号.

要在每个数据分组中增加一个检验和，并在分组到达接收方后检查它。如果检验和不正确，就说明分组损坏了，这个分组被悄无声息地丢弃掉。接收方的沉默对发送方来说就是一个信号，说明分组不是损坏了就是丢失了。发送方在每发送一个分组时要启动一个计时器。如果

确认号

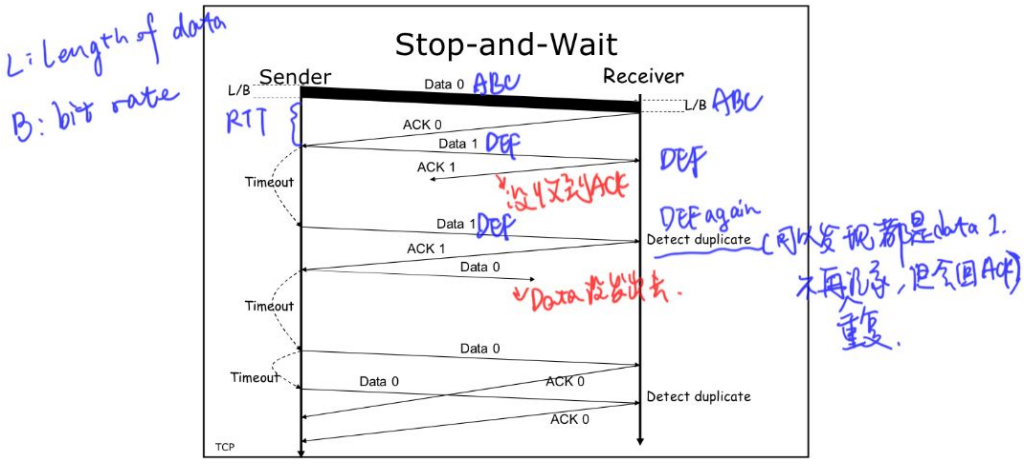
因为序号必须既适合数据分组，又适合确认，所以我们有以下约定：确认号总是声明了接收方准备接收的下一个分组的序号。例如，如果分组 0 已经安全完好地抵达了，那么接收方就发送一个确认号为 1 的 ACK（表示下一个希望接收分组 1）。如果分组 1 安全完好地抵达了，那么接收方就发送一个确认号为 0 的 ACK（表示下一个希望接收分组 0）。

序号指的是sequence no，确认号指的是ACK no。

[8] Whenever an entity accepts items from more than one source, it is referred to as multiplexing (many to one); whenever an entity delivers items to more than one source, it is referred to as demultiplexing (one to many).
[9] The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a segment.TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission. The segments are encapsulated in an IP datagram and transmitted.
Note that segments are not necessarily all the same size.
[10] The typical TCP header is 20 bytes, and the typical IPv4 header is also 20 bytes, so in this case overhead is TCP + IP =40 bytes
[11] A flow-control method in which each data unit must be acknowledged before the next one can be sent.

效率

如果我们的信道又粗又长，那么停止等待协议是非常低效率的。所谓粗是指我们的信道有较大的带宽（高数据率），而所谓长是指往返时延较长。它们俩的乘积被称为带宽时延积（bandwidth-delay product）。我们可以把信道看成是一根管道，那么带宽时延积就是管道的容量，以比特为单位。管道一直放在那，如果我们不充分利用它，我们的效率就很低。带宽时延积是对发送方在等待接收方确认的同时能够通过系统传送的比特数的度量。



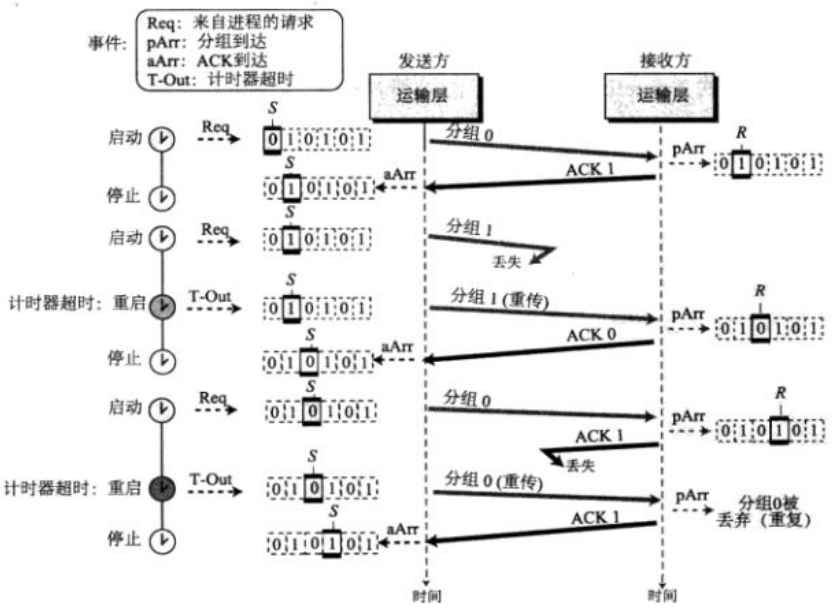
Stop-and-Wait Pros/Cons

- Simple operation
  - m=1 bit counter to detect duplicate data and ACK
  - No buffering needed in receiver
- Poor performance

  - Link bitrate: B
  - Packet length: L
  - Round trip time: RTT
  - Example: Stockholm-NY (1000bytes,10Mbps,100ms RTT)

$$U_{SnW} = \frac{L/B}{RTT + L/B}$$

$$U_{SnW} \approx \frac{10^3 * 8/10^7}{0.1 + 10^3 * 8/10^7} \approx 0.007$$



这种方法的缺点是利用率太低（尤其是带宽时延积较大时）。

②Go-Back-N

在收到ACK之前能够发送多个segment，但接收方只能缓存一个分组。发送方为发送出去的segment保存副本知道ACK送达。

序号

我们在前面曾经说过序号必须是模 2^m 的，其中 m 是序号字段的长度，以比特为单位。

模2^m即只能取0~2^m-1范围内的值。

计时器

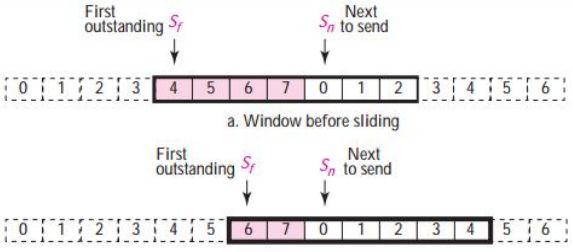
虽然每个发送出去的分组都可以有一个计时器，但在我们的协议中总共只使用了一个计时器。原因是第一个待确认的分组总是会最先超时。当这个计时器超时后，我们就重传所有待确认的分组。

重传分组

当计时器超时后，发送方重传所有待确认的分组。例如，假设发送方已经发送了分组 6（$S_n = 7$），但唯一的那个计时器超时了。如果 $S_f = 3$，那就是说分组 3、4、5 和 6 都尚未被确认，发送方返回到前面，重传分组 3、4、5 和 6。这就是为什么这个协议叫做返回 N 的原因。一旦超时，发送方就返回到 N 位置并重传所有分组。

滑动发送窗口[12]--窗口的最大值为2^m-1

Figure 13.24 *Sliding the send window*



发送窗口大小

我们现在可以来说明为什么发送窗口大小必须小于 2^m。作为一个例子，我们选择 $m = 2$，也就是说窗口大小可以是 $2^m - 1$，或者说 3。图 13.27 对一个大小为 3 的窗口和一个大小为 4 的窗口进行了比较。

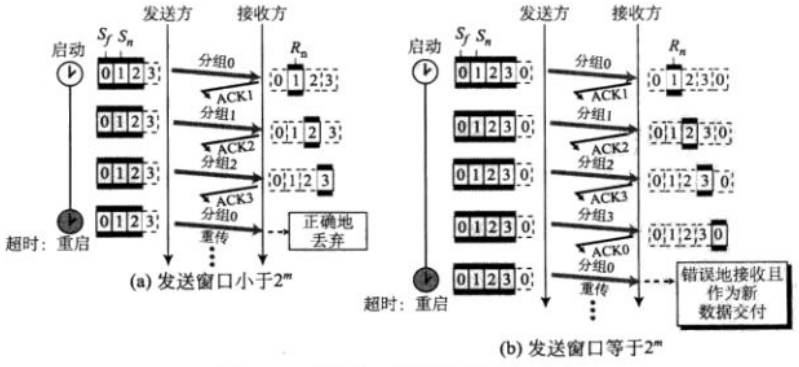


图 13.27　返回 N 协议的发送窗口大小

如果窗口大小是 3（小于 2^m）且所有三个确认都丢失了，那么唯一的那个计时器超时后所有三个分组被重传。此时接收方期盼的是分组 3 而不是分组 0，因此这些重复的分组到达后会被正确地丢弃掉。另一方面，如果窗口大小是 4（等于 2^m）且所有确认都丢失了，

[12] The send window is an abstract concept defining an imaginary box of maximum size = 2^m - 1 with three variables: Sf（第一个待确认的），Sn（下一个要发送的），and Ssize （窗口大小）

发送方会发送重复的分组 0。但是此时接收方的窗口正在期盼着接收分组 0（下一个循环），所以它就接受分组 0，它认为这不是一个重复分组，而是下一循环的第一个分组。这就是一个错误。从中我们可以看出发送窗口大小必须小于 2^m。

在返回 N 协议中，发送窗口大小必须小于 2^m，接收窗口大小始终为 1。

接收窗口

接收窗口保证了正确的数据分组被接收，且正确的确认分组被发送。在返回 N 协议中，接收窗口大小总是 1。接收方总是在期待着某一个特定分组的到达。任何失序到达的分组都会被丢弃并重传。图 13.25 所示为接收窗口。请注意我们只需要一个变量 R_n（接收窗口，

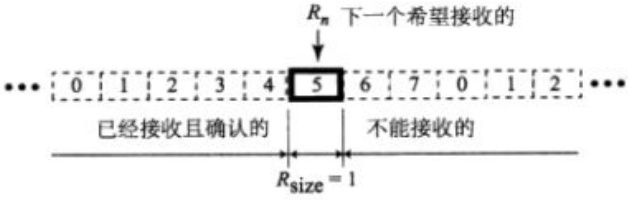
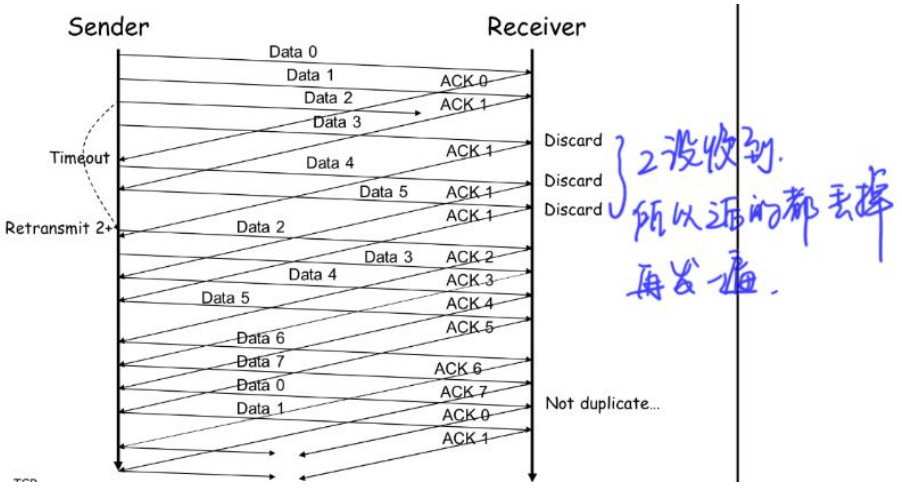


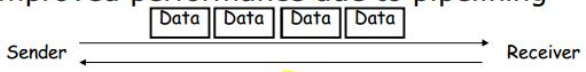
图 13.25　返回 N 的接收窗口



上图的例子m=3（计数器）.

Go-Back-N Pros/Cons

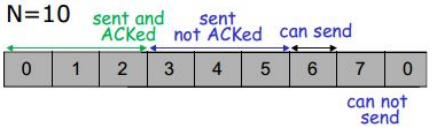
- Relatively simple operation
  - *m* bit counter to avoid duplicates
  - Sender window ($N \leq 2^m - 1$)
  - No buffering needed in receiver
- Improved performance due to pipelining



  - Link bitrate: B
  - Packet length: L
  - Round trip time: RTT
  - Example: Stockholm-NY, N=10

$$U_{GBN} = \min(1, \frac{N * L/B}{RTT + L/B})$$

$$U_{GBN} \approx \frac{10*10^3*8/10^7}{0.1+10^3*8/10^7} \approx 0.08$$



返回 N 与停止等待的比较

读者可能发现在返回 N 协议与停止等待协议之间有相似之处。停止等待协议实际上就是一种返回 N 协议，不过它只有两个序号且发送窗口大小为 1。换言之就是 $m=1$ 且 $2^m-1=1$。在返回 N 协议中，我们说计算都是模 2^m 的，在停止等待协议中计算都是模 2 的，

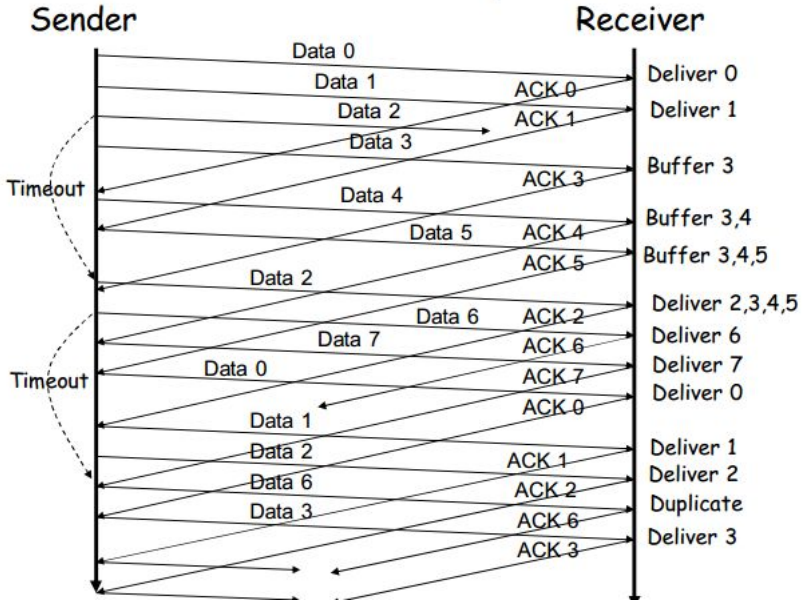
③Selective Repeat-N

窗口

选择重传协议也使用了两个窗口：一个发送窗口和一个接收窗口。但是这个协议中的两个窗口与返回 N 协议中的两个窗口有所不同。首先，它的发送窗口的最大值要小得多，是 2^{m-1}，原因我们稍后再讨论。其次，它的接收窗口与发送窗口一样大。

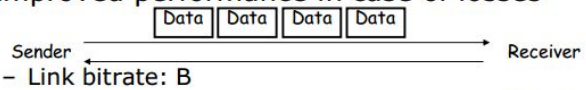
发送窗口的最大值可以是 2^{m-1}。例如，如果 $m=4$，那么序号就是从 0 到 15，但窗口的最大值只能是 8（在返回 N 协议中，这个值是 15）。我们在图 13.31 中描绘了选择重传协议的发送窗口以强调它的大小。

Selective Repeat-N



Selective Repeat Pros/Cons

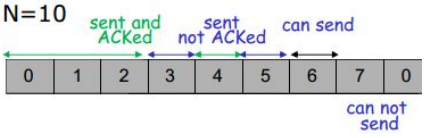
- Slightly more complex operation
  - m bit counter to avoid duplicates
  - Sender window ($N \le 2^{m-1}$)
  - Buffering needed in receiver
- Improved performance in case of losses



  - Link bitrate: B
  - Packet length: L
  - Round trip time: RTT
  - Example: Stockholm-NY, N=10

$$U_{SRN} = \min(1, \frac{N*L/B}{RTT+L/B})$$

$$U_{SRN} \approx \frac{10*10^3*8/10^7}{0.1+10^3*8/10^7} \approx 0.08$$



确认

这两个协议之间还存在其他一些区别。在 GBN 中，ackNo 是累计的，它定义了下一个希望接收的分组的序号，同时也证实了在此之前的所有分组都已经安全完好地被接收了。在 SR 中，确认的语义与此不同。在 SR 中，ackNo 只定义了安全完好地收到的那一个分组的序号，并不反馈任何其他分组的信息。

在选择重传协议中，确认号定义了无差错地接收到的那一个分组的序号。

例 13.9

假设发送方发送了 6 个分组：分组 0、1、2、3、4 和 5。发送方接收到一个 ackNo 为 3 的 ACK。如果这个系统使用的是 GBN 或者是 SR，它的含义各是什么？

解

如果系统使用的是 GBN，它就表示分组 0、1 和 2 都已被接收且无损坏，接收方现在期盼的是分组 3。如果系统使用的是 SR，它就表示分组 3 已经被接收且无损坏，这个分组没有说有关其他分组的任何事。

④三种flavor总结

	发送窗口（最大值）	接收窗口（最大值）	序号	计时器（timer）
Stop-and-wait	1	1	0~1	
Go-Back-N	2^m-1	1	0~2^m-1	所有待确认的segment共用一个计时器
Selective Repeat-N	2^{m-1}	2^{m-1}	0~2^m-1	每个待确认的segment各一个单独的计时器

▵Retransmission Time-Out (RTO)– Time to wait for the ACK of a segment（不是一个fixed number）

Calculation of the RTO

- Method used today (RFC 2988, Van Jacobson '88)
  - Update estimated RTT variation

$$RTTvar[k+1] = (1-\beta)RTTvar[k] + \beta\,|sRTT[k] - mRTT[k]|$$

  - Update estimated RTT mean

smooth RTT　　measure RTT，每次实际测量的结果

$$sRTT[k+1] = \alpha\, sRTT[k] + (1-\alpha)\, mRTT[k]$$

  - Update RTO based on estimated mean *and* variation

$$RTO = sRTT[k+1] + 4RTTvar[k+1]$$

- How do you measure *mRTT*?
  - Time between segment is sent and ACK is received (?)

▵Karn's Algorithm

- Scenario
  - Segment not ACKed and is retransmitted
  - When ACK is received the sender does not know if
    - ACK was sent for the original segment
    - ACK was sent for the retransmitted segment
- Solutions
  - Use the TCP timestamp option (see later)
  - Karn's algorithm　不考虑重传的segment的RTT
    - Don't consider the RTT of a retransmitted segment
    - Don't update RTT value until you get an ACK without need for retransmission　只有当不需要重传就能ACK时才更新RTT

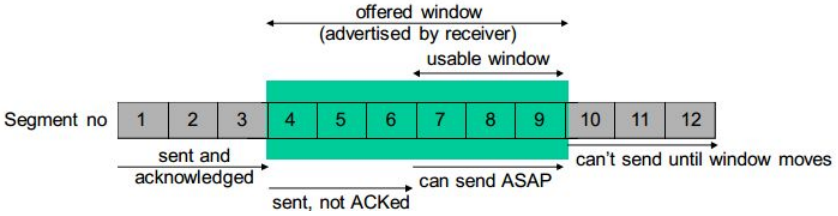
4) Flow control -- Adapt to the receiver's capabilities

• Ensure that receiver does not get overwhelmed with data sent by the sender

• TCP uses a sliding window protocol

TCP Sliding Windows

- Receiver: *offered window* (sent with ACK)
  - data that it can receive beyond what it ACKed
  - can send an ACK with an offered window of 0!
    - later it sends *window update* with offered window size > 0
- Sender: *usable window*
  - data it can send immediately



△Delayed acknowledges[13]

△Persistence Timer

为了处理零窗口值的通告，TCP 需要另一个计时器。如果接收 TCP 宣布窗口值为零，那么发送 TCP 就会停止发送报文段，直至接收 TCP 发送来一个宣布窗口大小非零的确认。但是这个 ACK 报文段有可能会丢失。应当记住，在 TCP 中的确认报文段既不需要确认，也不需要重传。若确认报文段丢失了，接收 TCP 仍然认为这个确认已经完成了任务，并等待着发送 TCP 发送下面的报文段。对于仅含有一个确认的报文段是没有重传计时器的。发送 TCP 由于没有收到确认，就等待对方发送一个确认来通知窗口的大小。这两个 TCP 都在永远地等待着对方（死锁）。

为了纠正这个死锁问题，TCP 为每个连接使用一个**持续计时器**（persistence timer）。当发送 TCP 接收到窗口值为零的确认时，就启动一个持续计时器。当持续计时器超时后，发送 TCP 就发送一个特殊的报文段，称为探测报文段。这个报文段只有 1 个字节的新数据。它有序号，但它的序号永远不需要确认，甚至在计算剩余数据的序号时，这个序号也被忽略。探测报文段促使接收 TCP 重传一个确认。

Periodically sends window probes to find out if window size has increased, e.g., every 60 seconds

持续计时器的时间长度被设置为重传时间的值。但是，若没有收到从接收方发来的响应，则需发送另一个探测报文段，并将其持续计时器的值加倍，且该计时器复位。发送方继续发送探测报文段，不断地将持续计时器的值加倍和复位，直到这个值达到一个门限（通常是 60 秒）为止。在这以后，发送方每隔 60 秒就发送一个探测报文段，直到窗口重新打开。

△Silly Window Syndrome 的解决办法

在滑动窗口的操作中可能出现一个严重的问题，这就是发送应用程序产生数据的速度很慢，或者接收应用程序消耗数据的速度很慢，或者两者都有。不管是上述情况中的哪一种，都会使得发送数据的报文段很小，这就会降低运行的效率。例如，假若 TCP 发送的报文段只包含 1 个字节的数据，那么意味着我们为传送 1 字节的数据而发送了 41 字节的数据报（20 字节的 TCP 首部和 20 字节的 IP 首部）。此时的额外开销达到了 41/1，这就表示我们使用网络容量的效率非常低。再算上数据链路层和物理层的额外开销后，这种低效率的程度就更加严重了。这个问题称为**糊涂窗口综合征**（silly window syndrome）。我们先来分

[13] Advantages:
1. ACK traffic is reduced 2. Increased chance that data can be piggy-backed（一方发送sequence时，packet内同时包含了对另一方的ACK） on the ACK

发送方慢：

解决这个问题的方法是防止发送 TCP 一个字节一个字节地发送数据。必须要强迫发送 TCP 等待，并把数据收集成较大的数据块后再发送。发送 TCP 需要等待多长时间呢？如果

Nagle 算法 **Nagle 算法**非常简单：

1．发送 TCP 把它从发送应用程序收到的第一块数据发送出去，哪怕只有 1 个字节。

2．在发送了第一个报文段后，发送 TCP 就在输出缓存中累积数据并等待，直至收到接收 TCP 发来的确认，或者已积累了足够的数据可以装成最大长度的报文段。这时，发送 TCP 就可以发送这个报文段了。

3．对剩下的传输，不断重复步骤 2。如果收到了对报文段 2 的确认，或者已积累到足够多的数据可以装成最大长度的报文段，报文段 3 就必须发送出去。

Nagle 算法的巧妙之处在于它的简单，并且实际上它权衡了应用程序产生数据的速率和网络运输数据的速率。若应用程序比网络更快，则报文段就较大（最大长度报文段）。若应用程序比网络慢，则报文段就较小（小于最大长度报文段）。

Solution: Nagle's Algorithm

- TCP connection can have only one unacknowledged tinygram
  - Data are accumulated while waiting accumulate累积
  - Sent as one segment when the ACK arrives, or when maximum segment size can be filled
- The faster ACKs come, the more often data are sent
  - On slow WANs fewer segments are sent than on fast LANs
  - Rarely invoked on a LAN
    - Round trip time (RTT) ~1ms - to generate data faster than this would require typing faster than 60 characters per second!

接收方慢：

Clark 解决方法 **Clark 解决方法**是只要有数据到达就发送确认，但在缓存中有足够大的空间放入最大长度的报文段之前，或者至少有一半的缓存空间为空之前，一直都宣布窗口大小为零。

推迟确认 第二种解决方法是推迟发送确认。这就表示当报文段到达时并不立即发送确认。接收方在对收到的报文段进行确认之前一直等待，直至输入缓存有足够的空间为止。推迟发送确认防止了发送 TCP 滑动它的窗口。发送 TCP 在发送了数据之后就停下来，因而防止了这种征状的出现。

推迟确认还有另外一个优点：它减少了通信量。接收方不需要对每一个报文段进行确认。但它也有一个缺点，就是确认的推迟有可能迫使发送方重传未被确认的报文段。

TCP 对这个优点和缺点进行了平衡。目前它的定义是推迟确认不能超过 500 ms。

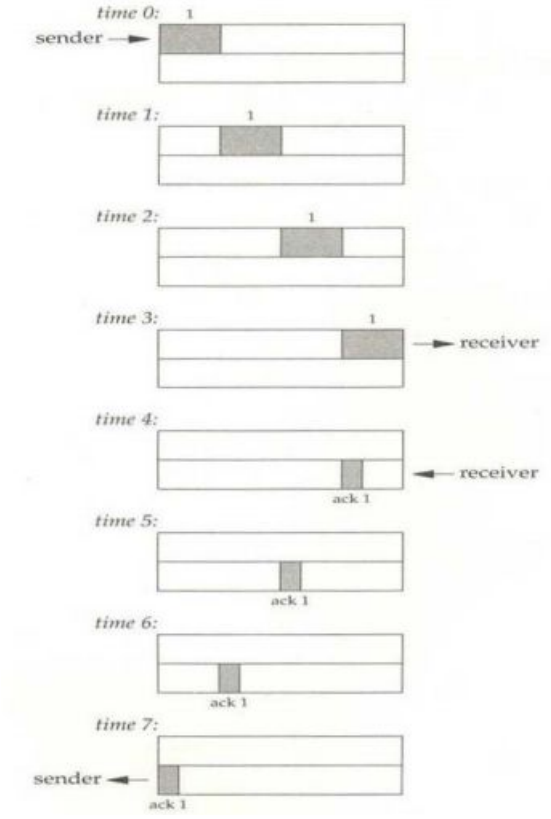
△Bandwidth-Delay Product

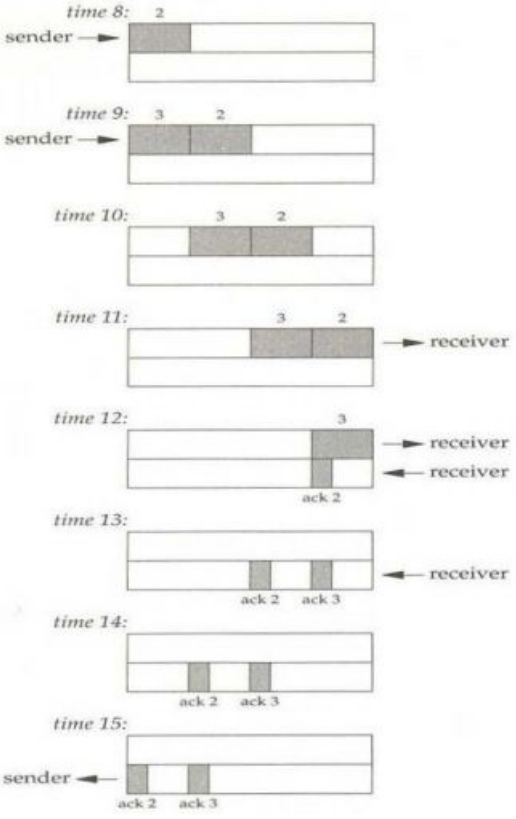
• The ”capacity” of the ”pipe”

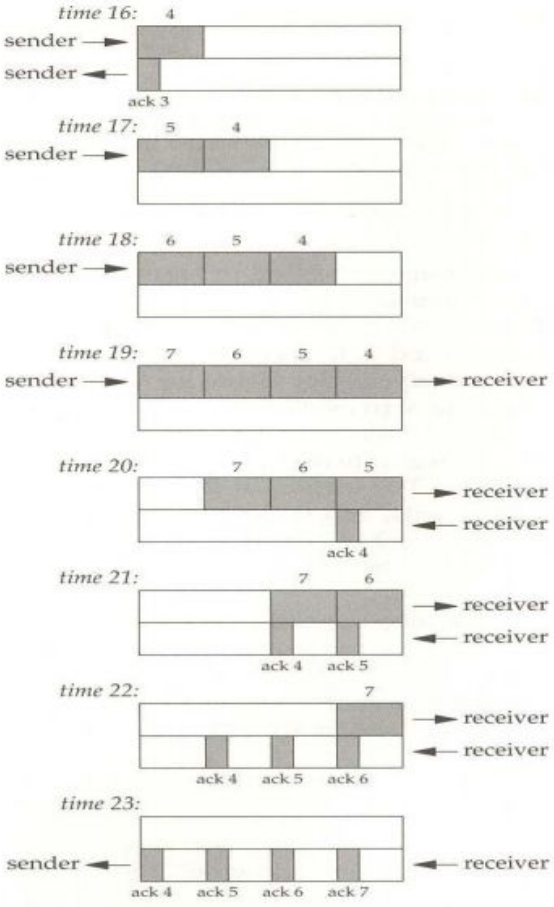
capacity(bits) = bandwidth(bits/sec) x RTT(sec)

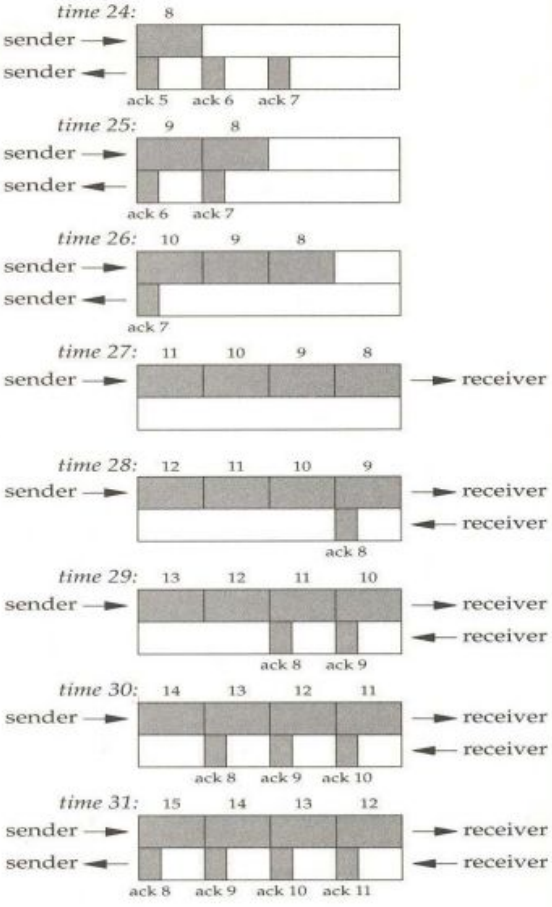
• The receiver advertised window should be higher

△TCP Bulk Data Flow


time 0:
1
sender
time 1:
1
time 2:
1
time 3:
1
receiver
time 4:
receiver
ack 1
time 5:
ack 1
time 6:
ack 1
time 7:
sender
ack 1


time 8:
2
sender
time 9:
3
2
sender
time 10:
3
2
time 11:
3
2
receiver
time 12:
3
receiver
receiver
ack 2
time 13:
receiver
ack 2
ack 3
time 14:
ack 2
ack 3
time 15:
sender
ack 2
ack 3


time 16:
4
sender
sender
ack 3
time 17:
5
4
sender
time 18:
6
5
4
sender
time 19:
7
6
5
4
sender
receiver
time 20:
7
6
5
receiver
receiver
ack 4
time 21:
7
6
receiver
receiver
ack 4
ack 5
time 22:
7
receiver
receiver
ack 4
ack 5
ack 6
time 23:
sender
receiver
ack 4
ack 5
ack 6
ack 7

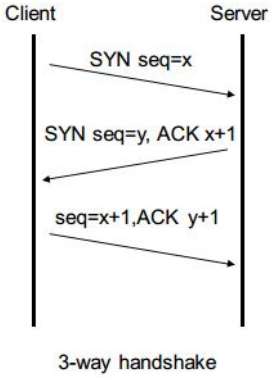

time 24:
8
sender
sender
ack 5
ack 6
ack 7
time 25:
9
8
sender
sender
ack 6
ack 7
time 26:
10
9
8
sender
sender
ack 7
time 27:
11
10
9
8
sender
receiver
time 28:
12
11
10
9
sender
receiver
receiver
ack 8
time 29:
13
12
11
10
sender
receiver
receiver
ack 8
ack 9
time 30:
14
13
12
11
sender
receiver
receiver
ack 8
ack 9
ack 10
time 31:
15
14
13
12
sender
sender
receiver
receiver
ack 8
ack 9
ack 10
ack 11

5) Connection Management--Establishment/tear down

▵TCP connection establishment

TCP Connection Establishment

TCP uses a three-way handshake to establish a connection



3-way handshake:

- Guarantees both sides ready to transfer data
- Allows both sides to agree on initial sequence numbers, MSS, window size

Initial sequence number (ISN) must be chosen so that each incarnation of a specific TCP connection between two end-points has a different ISN.

Normally, client performs *active open*, server performs *passive ope*
Alternative: simultaneous open

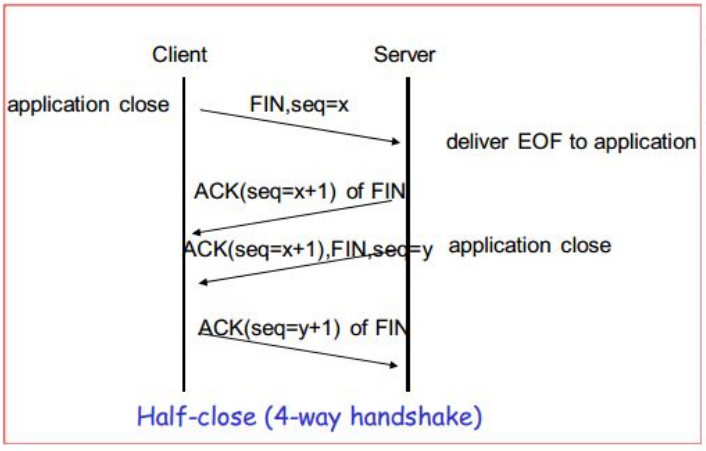
▵Keepalive Timer--Avoid TCP connections to exist forever

在某些实现中要使用**保活计时器**（keepalive timer）来防止两个 TCP 之间的连接长时间空闲。假定一个客户打开了到服务器的一条连接，传送过一些数据，然后就变得静默了。也许是因为这个客户出了什么故障。在这种情况下，这个连接将永远处于打开状态。

为了解决这个问题，绝大多数的实现都是给服务器设置了一个保活计时器。每当服务器收到客户的信息，就把该计时器复位。超时通常设置为两小时。若服务器过了两小时还没有收到客户的任何信息，它就发送一个探测报文段。若连续发送了 10 个探测报文段（每隔 75 秒一个）还没有收到响应，它就假定客户出了故障，并终止这个连接。

▵TCP connection teardown

TCP Connection Teardown



4 segments needed to terminate a connection

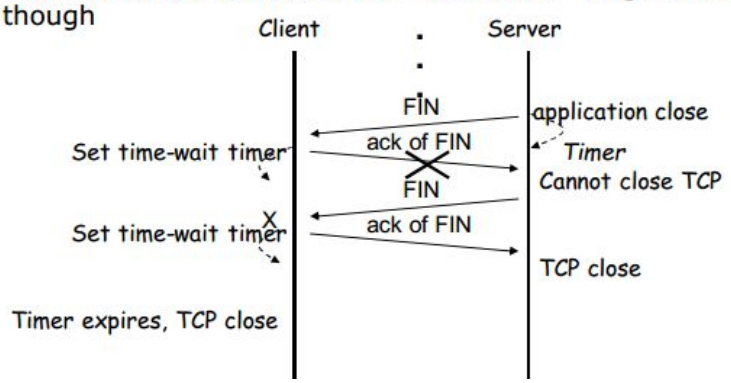
3-way handshake is allowed as well

Simultaneous close looks the same

Normally, client performs *active close* and server performs *passive close*

▵Time-Wait Timer--Connection termination

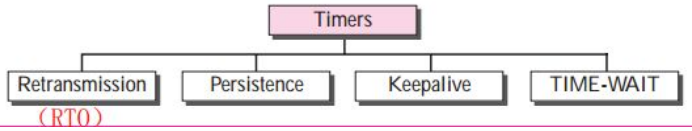
- The connection is held in TIME-WAIT state
  - allows duplicate FIN segments to arrive at the destination (send ACK)
- TWT is usually twice the maximum lifetime of a segment
  - Maximum Segment Lifetime = 0,5...2 mins (←IP TTL limit)
- Port should not be reused within this time – might be reopened though



14

汇总：TCP的各种计时器

Figure 15.38 *TCP timers*



6) Congestion control-- Adapt to network conditions

The window size must depend on the network's state as well!

- Congestion Window maintained by the sender
  - Updated in response to losses and/or delay
- Sender maintains 2 window sizes:
  - Receiver-advertised window (RWND)
  - Congestion window (CWND)
- Actual window size = min(RWND, CWND)
- Two phases
  - Slow start
  - Congestion avoidance → Additive increase Multiplicative decrease (AIMD)

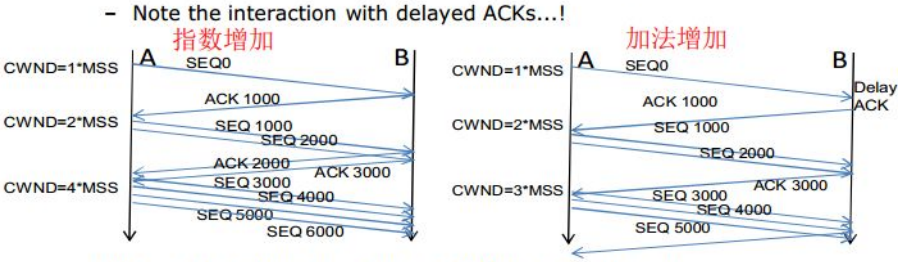
(RWND) CWND

发送方有两种信息：接收方通告的窗口大小和拥塞窗口大小

TCP 处理拥塞的一般策略是基于三个阶段：慢开始、拥塞避免和拥塞检测。在慢开始阶段，发送方从非常慢的传输速率开始，但很快就把速率增大到一个门限值。当到达门限后，数据率的增长开始放慢。最后，只要一检测到拥塞，发送方就又回到慢开始或者是拥塞避免阶段，具体取决于拥塞是如何检测到的。

[14] FIN是一种TCP的flag， which means sender has finished sending data

- All TCP implementations *must* implement slow start
- Operation
  - At beginning of connection, CWND ⇐ [1..4] x MSS 第一阶段
  - For each ACK: `CWND += min(N,MSS)` 第二阶段
    - N is the new data (in bytes) acknowledged by ACK
- Not slow: window size grows exponentially
  - Note the interaction with delayed ACKs...!



- Stop using *slow start* when CWND > *ssthresh*

△Slow Start & Congestion Avoidance

发送方从 cwnd = 1 MSS 开始。这表示发送方只能发送一个报文段。在第一个 ACK 到达后，拥塞窗口大小增加 1，也就是说现在 cwnd 是 2。此时可以发送两个报文段。当相应的两个 ACK 到达后，对于每一个 ACK，窗口大小增加 1 MSS，这就表示 cwnd 现在是 4。此时可以发送 4 个报文段。当四个 ACK 到达后，窗口大小又增加了 4，也就是 cwnd 等于 8。

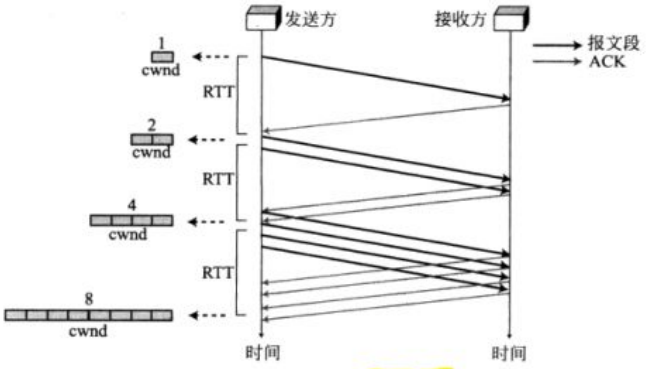


图 15.34 慢开始，指数增大

慢开始不能无限制地继续下去。一定要有一个门限来停止这个阶段。发送方密切关注一个称为 ssthresh（慢开始门限）的变量。当以字节计的窗口大小到达这个门限时，慢开始阶段就结束了，并进入下一个阶段。

在慢开始算法中，拥塞窗口大小按指数规律增长，直至达到门限为止。

拥塞避免：加法增大

如果我们从慢开始算法开始，拥塞窗口大小就按指数规律增长。为了在拥塞发生之前避免它，就必须使这种指数增长变慢。TCP 定义了另一个算法，叫做**拥塞避免**（congestion

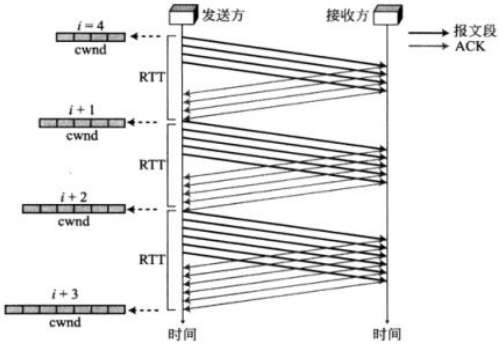


图 15.35 拥塞避免，加法增大

▵TCP Header
20 byte 固定 + 20 byte optional
TCP中的checksum是必须的（UDP不是）
Reading instruction:
Ch 13,14,15.1-15.4

Lecture 10 &11 Application Layer

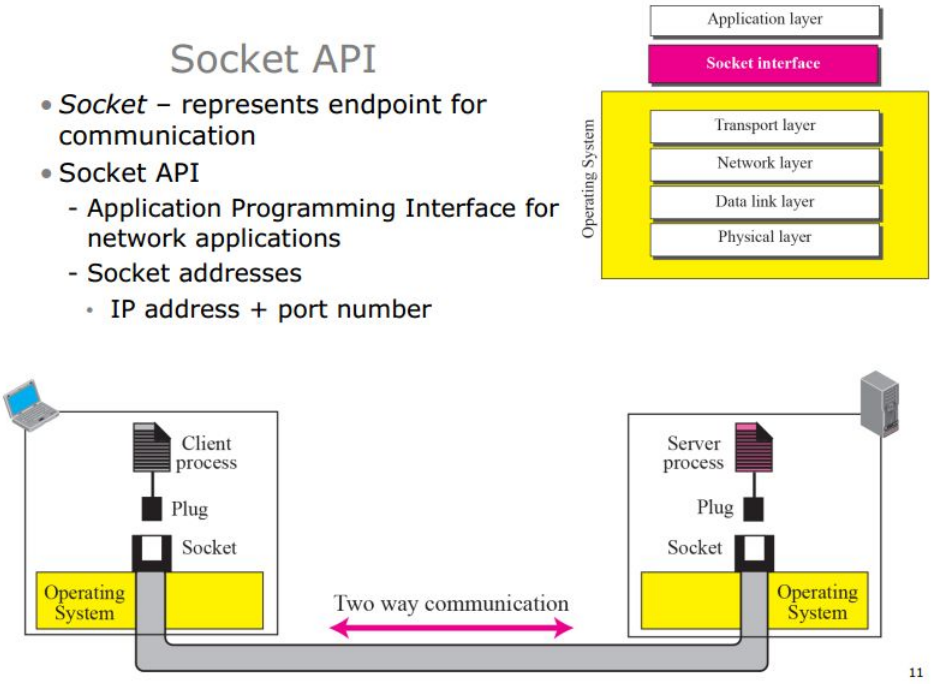
1.In general

Applications run on end-systems only
Possible structure of applications:
- Client-server

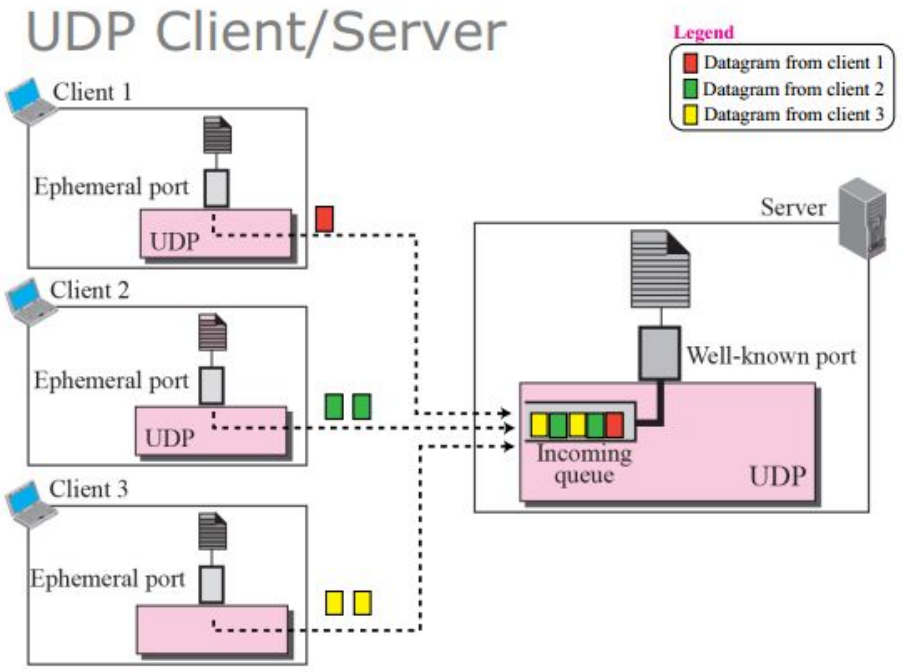
Server	**Client**
Always on	May be intermittently connected
At a permanent, well-known location (For instance, an HTTP server is at port 80, by default)	May have dynamic IP addresses
Can service many clients	“Ephemeral” ports (Short-lived, dynamically allocated ports)

- Peer-to-peer (P2P)
• No always-on server
• Peers request service from other peers, provide service in return to other peers

2.Creating network applications

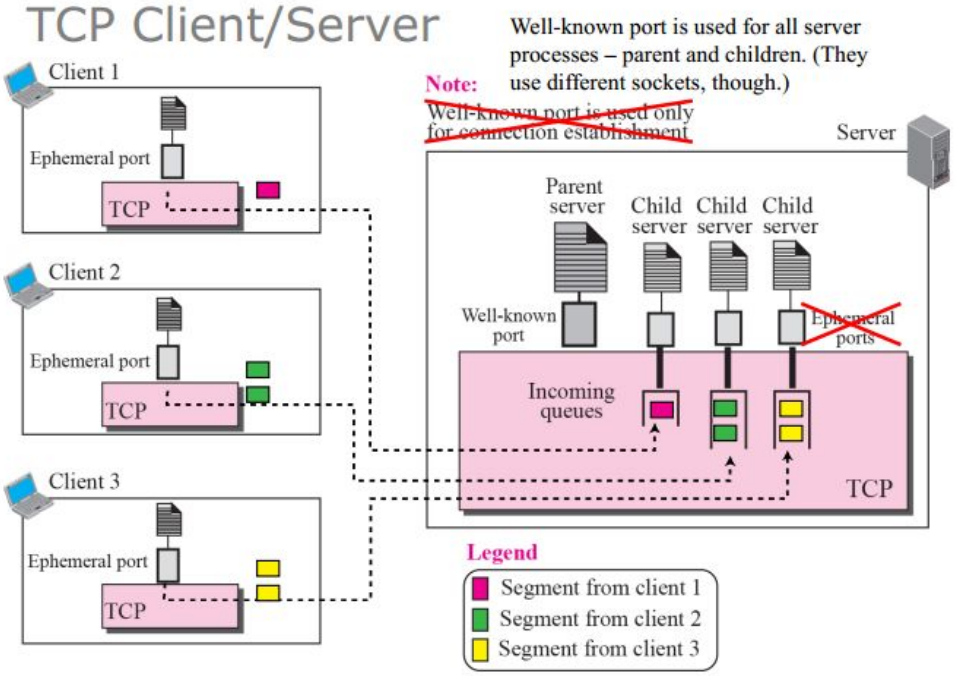


Sequencial Server



- Sequential server
  - Process one client request at a time

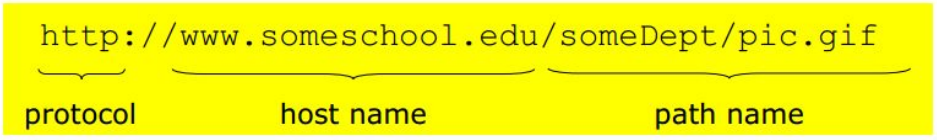
Concurrent Server



- Concurrent server
  - Handle multiple clients at the same time

3.Web and HTTP

- Web page consists of *base HTML-file* which includes *several referenced objects*
- Each object is addressable by a *URL, Uniform Resource Locator,* e.g., 可以通过URL访问



HTTP: hypertext transfer protocol--Web application layer protocol

△client/server model (client: browser, server: Web server)

△uses TCP，步骤如下:

•client initiates TCP connection (creates socket) to server, port 80

•server accepts TCP connection from client

•HTTP messages (application layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)

•TCP connection closed

△HTTP is stateless

•Request/response

•Server maintains no information about past client requests

△HTTP Response Status Codes

Status code appears in first line in server-to-client response message. Some sample codes:

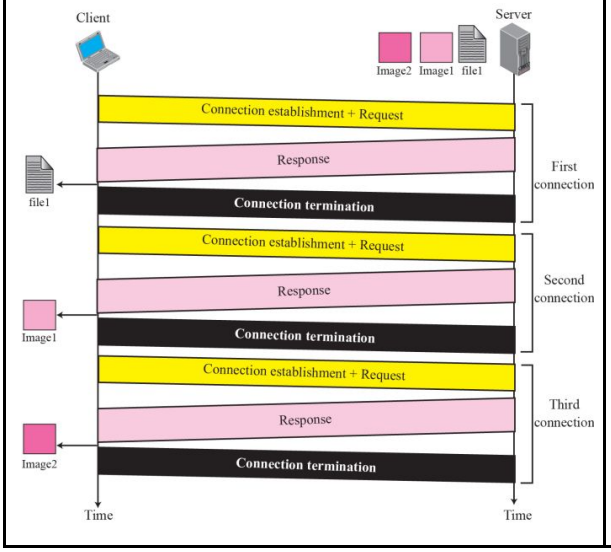
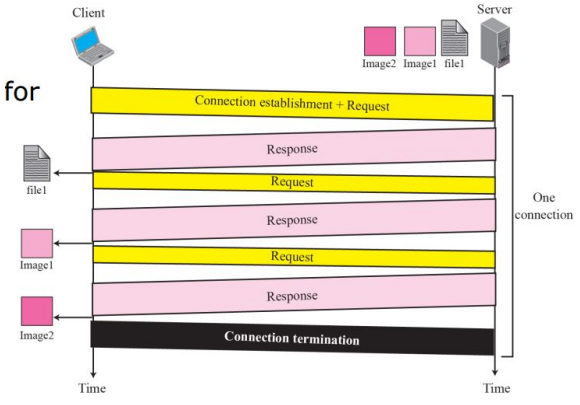
200 OK - request succeeded, requested object later in this response

301 Moved Permanently - requested object moved, new location specified later in this response (Location:)

400 Bad Request - Request not understood by server

404 Not Found - requested document not found on this server 505 HTTP Version Not Supported

△TCP connection strategies

Non-persistent Connection	Persistent Connection
	

One TCP connection per HTTP transaction	1. Reuse same TCP connection for multiple HTTP transactions - Default as of HTTP 1.1 2. How long should connection be left open? - Occupies server resources - Controlled by “Keep-Alive” header

▵HTTP cookies

What cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)
- tracking – sites you visit

How to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

aside

cookies and privacy:

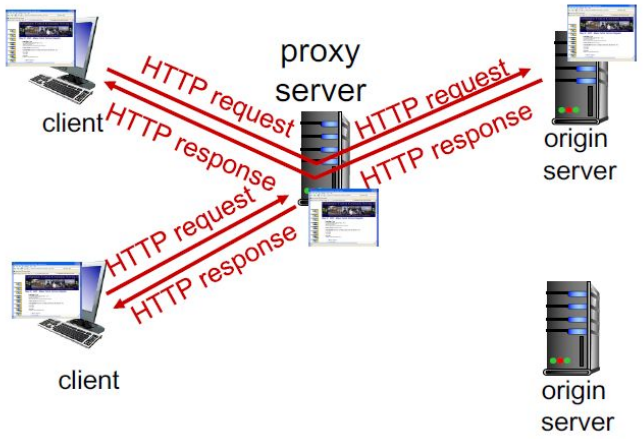
- cookies permit sites to learn a lot about you
- For instance, you may supply name and e-mail to sites

aside

Controlling cookies:

- Cookie acceptance policy in browser
- List and remove cookies manually

▵Web caches (proxy server) 代理服务器



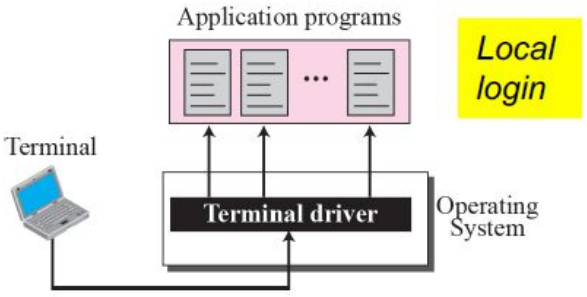
- Cache acts as both client and server
  - server for original requesting client
  - client to origin server
- Typically cache is installed by ISP (university, company, residential ISP)
- Browsers also have built-in caches

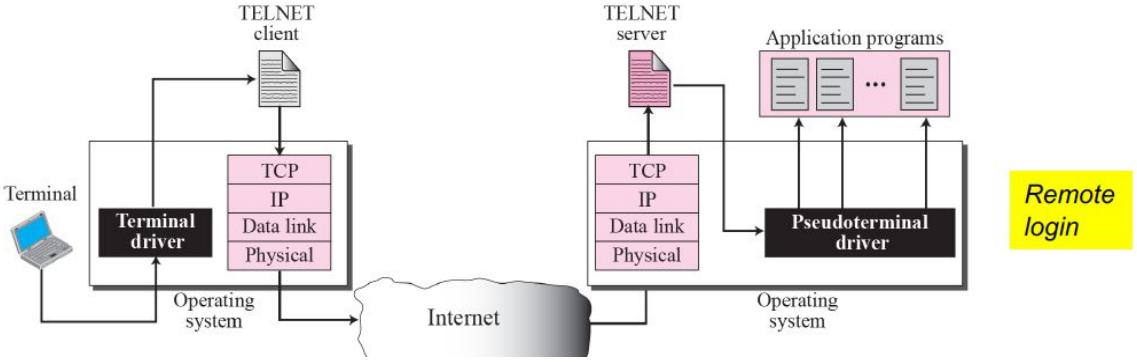
Why Web caching?

- Reduce response time for client request
- Reduce traffic on an organization’s access link
- Internet dense with caches: enables ”poor” content providers to effectively deliver content (so too does P2P file sharing)

4. Remote login--Telnet & SSH

1) Telnet Remote Login





2) SSH – Secure Shell

• Telnet considered insecure

- No encryption – eavesdropping
- No authentication of client/server

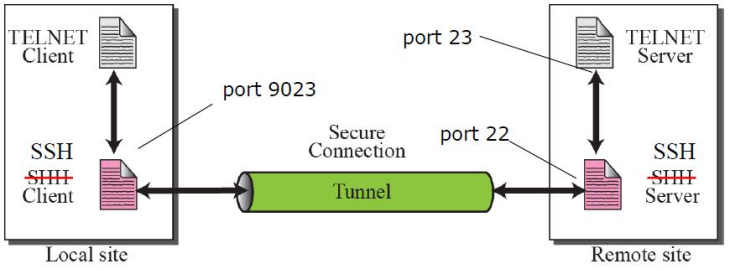
• SSH

- Encryption and authentication
- Create a secure (encrypted and authenticated) channel over TCP
- Default port 22

3) Port Forwarding

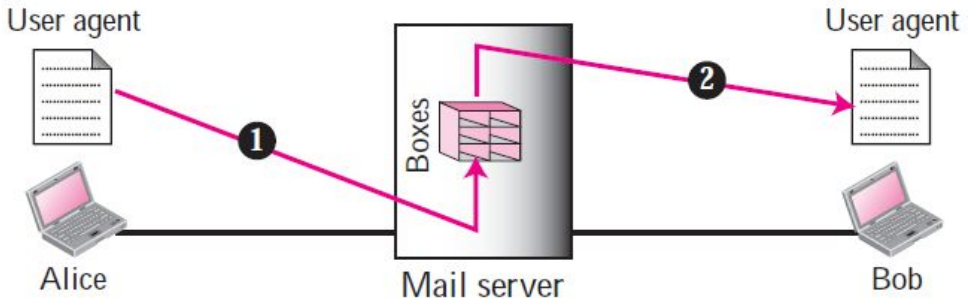
- Create a secure connection, *tunnel*, between TCP ports on two machines
- SSH client/server act as *proxies*
- In this way, any legacy (insecure) application can run over a secure connection

  - For example, telnet over a secure tunnel:
    - Set up SSH tunnel with port forwarding from port 9023 on "Client" to port 23 (telnet) on "Server"
    - Command "telnet localhost 9023" on "Client" will connect to telnet server on port 23 on "Server" via SSH tunnel

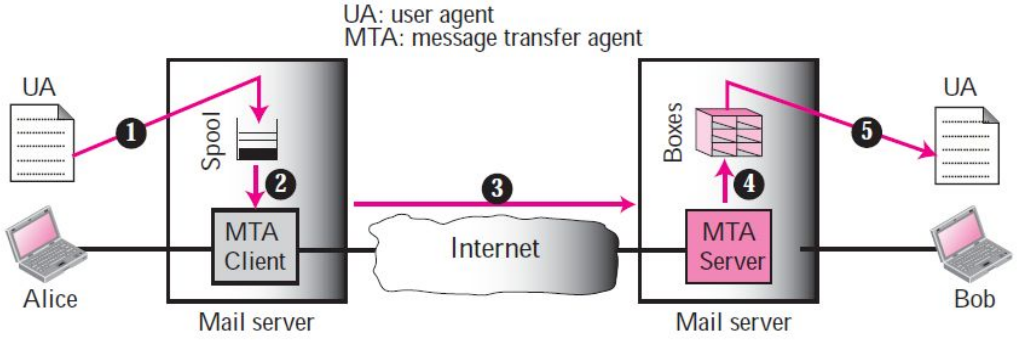


5. Email

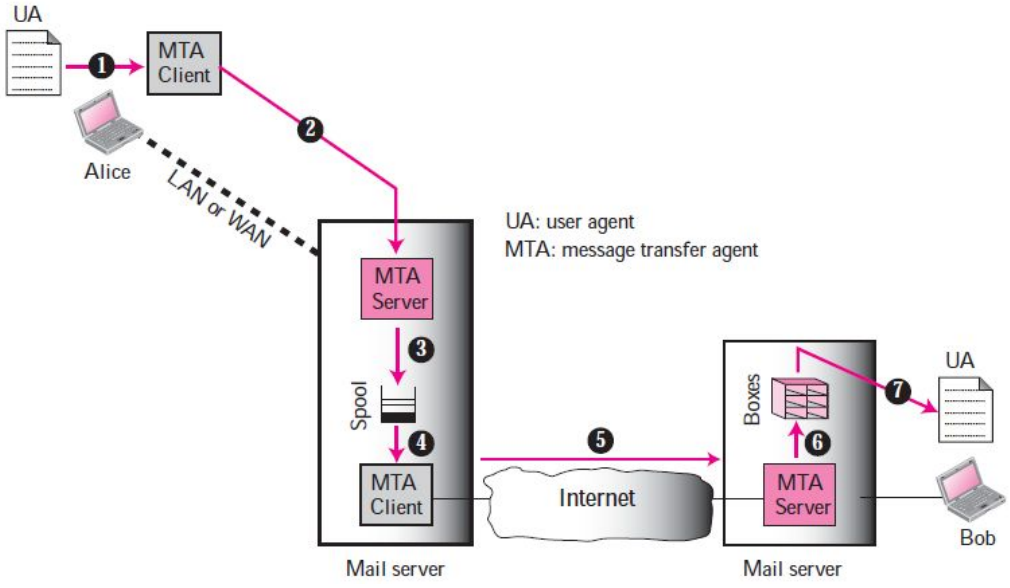
Case1. When the sender and the receiver of an e-mail are on the same mail server, we need only two **user agents**[15].



Case2. When the sender and the receiver of an e-mail are on different mail servers, we need **two UAs and a pair of MTAs** (client and server).



Case3. When the sender is connected to the mail server via a LAN or a WAN, we need **two UAs and two pairs of MTAs** (client and server).



Case4. When both sender and receiver are connected to the mail server via a LAN or a WAN, we need **two UAs, two pairs of MTAs (client and server), and a pair of MAAs** (client and server). This is the most common situation today.

---

[15] User Agent--Program to create and read e-mail
- Examples: Outlook, OS X Mail, Thunderbird, Kmail,Envelope, ...

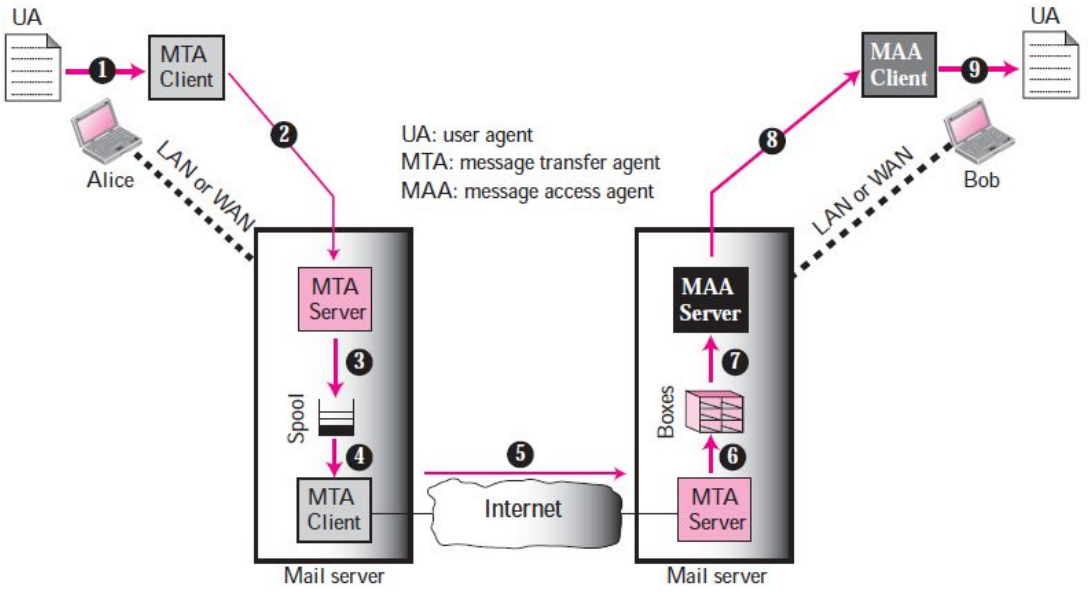
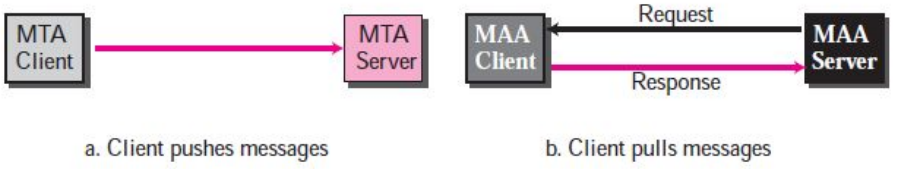


Figure 23.5 *Push vs. pull*

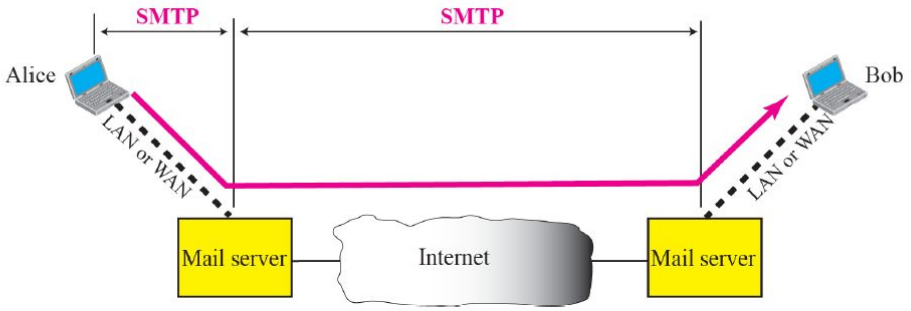


Push和Pull用不同的协议：

Push--SMTP, Pull--POP, IMAP

a. SMTP

Simple Mail Transfer Protocol (SMTP)



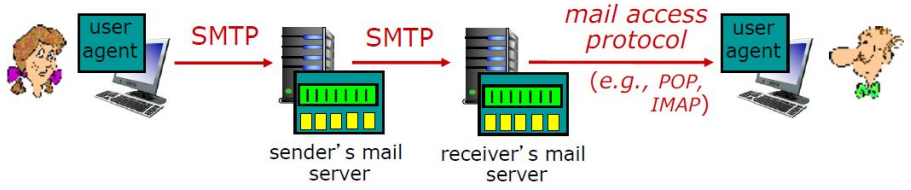
- Uses TCP to reliably transfer email message from client to server
  - port 25
- Direct transfer: sending server to receiving server
- Three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - Closure
- Command/response interaction (like HTTP, FTP)
  - commands: ASCII text
  - response: status code and phrase
- Messages must be in 7-bit ASCII

SMTP requires message(header & body) to be in 7-bit ASCII

comparison with HTTP:

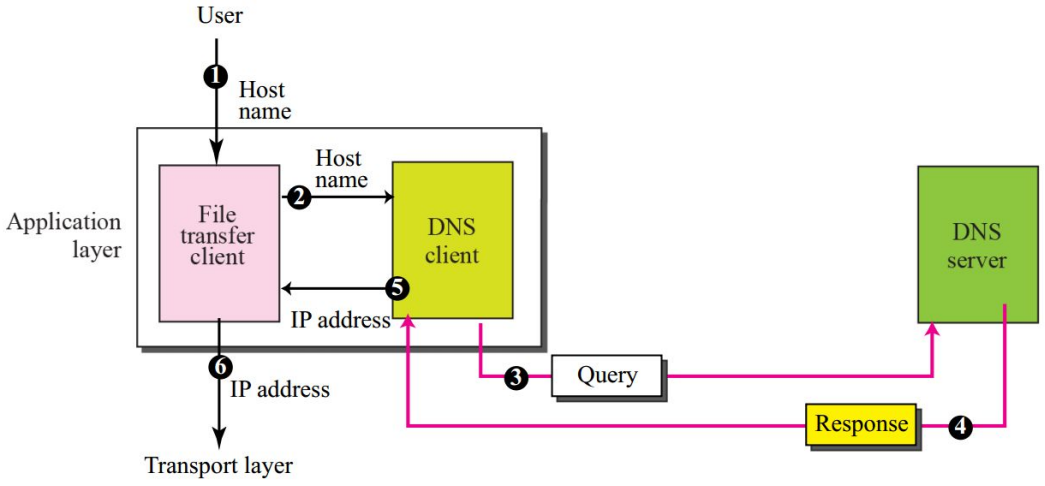
- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response message
- SMTP: multiple objects sent in multipart message

b. Mail Access Protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored messages on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

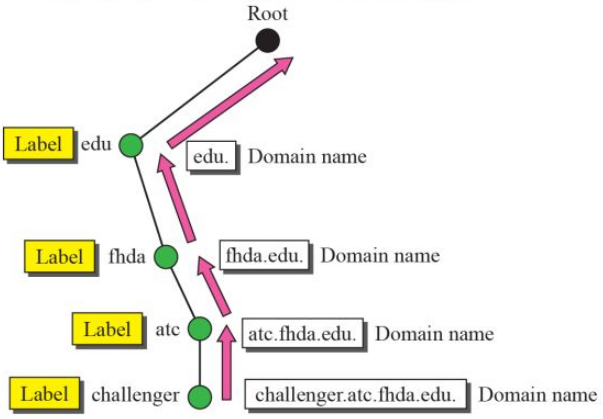
Lecture 12 DNS



- One name can map to several addresses
- One address can have several names

Domain Names and Labels

- Nodes have labels
  - Root's label is empty string
- Each node represents a domain name



- Domain name is sequence of labels separated by dots "."
- A full domain name is a sequence from bottom to top
  - Since root's label is the empty string, a full domain name ends with a dot "."
  - Fully Qualified Domain Name (FQDN)
- Otherwise partial name
  - Partially Qualified Domain Name (PQDN)
    - Term seldom used in practice though
  - Relative to a node in the tree

FQDN	PQDN
challenger.atc.fhda.edu. cs.hmme.com. www.funny.int.	challenger.atc.fhda.edu cs.hmme www

- Distributed database organized as a tree of name servers

Client wants IP for www.amazon.com; 1st approximation:

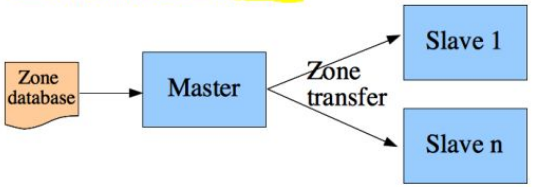
- client queries a root server to find "com" TLD DNS server
- client queries "com" TLD DNS server to get "amazon.com" DNS server
- client queries "amazon.com" DNS server to get IP address for "www.amazon.com"

15[16]

[16] TLD: 顶级域（或顶级域名；英语：Top-level Domain；英文缩写：TLD）是互联网DNS等级之中的最高级的域，它保存于DNS根域的名字空间中。顶级域名是域名的最后一个部分，即是域名最后一点之后的字母，例如在example.com这个域名中，顶级域是.com（或.COM），大小写视为相同。

Master and Slaves

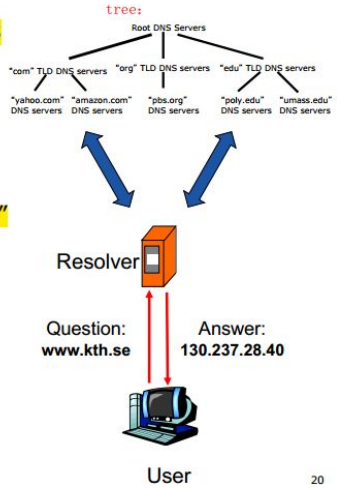
- One or several name servers are *authoritative name servers* for a zone
  - Responsible for that part of the namespace
- One authoritative server is master (primary server)
  - Has master copy of zone file
- Other authoritative servers are slaves (secondary servers)
  - Redundancy!
- Zone file changes distributed from master to slaves
  - "Zone transfer" over TCP



17

Resolution – Making DNS Queries

- "Resolver" performs DNS lookups on behalf of users (clients)
  - Sends queries to servers
    - Root, TLD, authoritative servers, etc
  - Maintains a cache of recent responses
  - Both DNS client and server – "proxy"
- A.k.a. "Default name server", "resolving name server", "local name server", …
- Part of IP configuration of a host
  - DHCP configuration
  - Which resolver are you using right now?

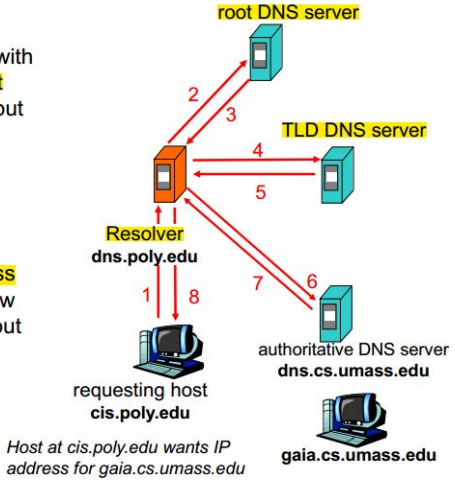


20

Recursive and Iterative Resolution

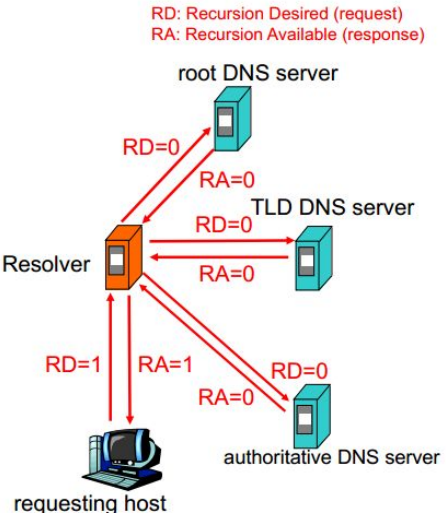
Iterative resolution:
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"
  - What root and TLD servers do

Recursive resolution:
- Contacted server replies with the requested address
- If the server does not know the address, it will find it out
  - What resolver does

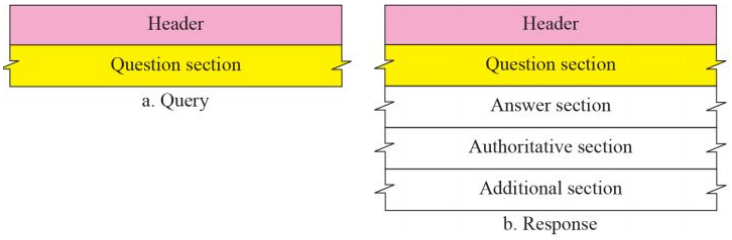


[17] Having multiple name servers is for redundancy. When an organization has multiple name servers, there is one primary server and several secondary servers. The zone file is updated on the primary server, and then the updates are transferred automatically to the secondaries through "zone transfers".

• If client requests recursion, and server agrees, the server resolves the name for the client
--Through iterative resolution
• Otherwise server sends back whatever information it has about the name
--Typically name of server to contact (but not necessarily)
• Normally, only resolvers agree to recursion



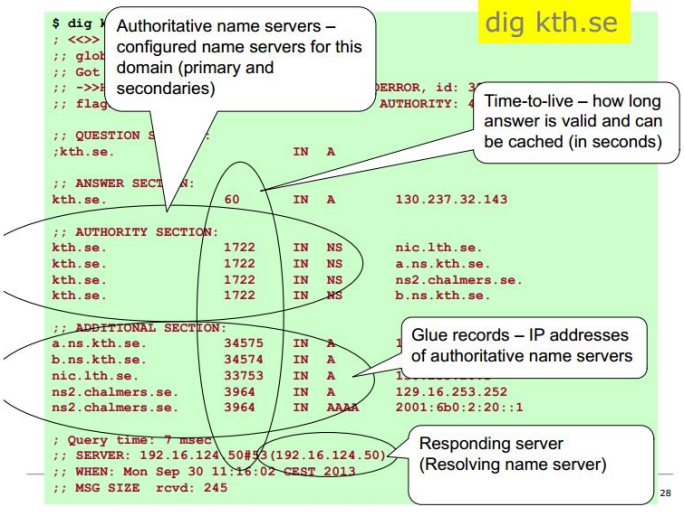
DNS Query and Response

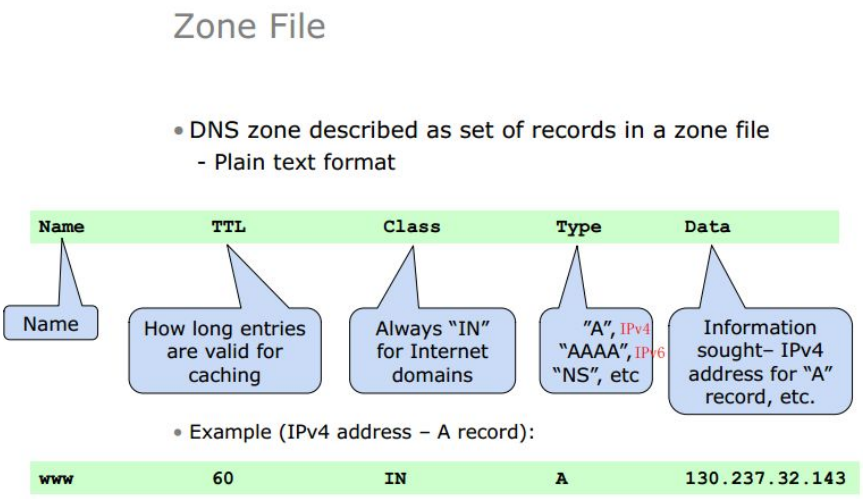


- UDP and TCP port 53 (by default)
- Query-response protocol to get resource records from the DNS database

Side note DNS primarily uses UDP. Messages are getting larger due to new functionality being introduced, such as security, so a single IP datagram might not be enough to carry a DNS message. Then TCP might be a better choice, compared to IP fragmentation. Zone transfers always use TCP.

Quearying tool: dig (domain information groper)





Summary

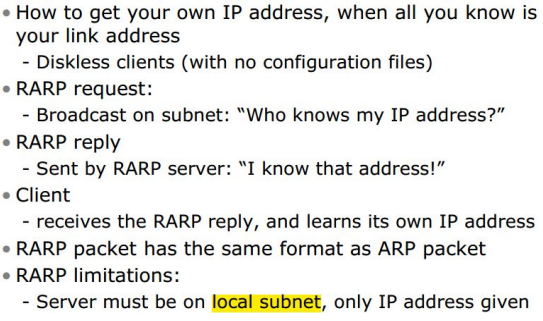
• Domain name space organized in hierarchy
- Generic domains, country domains, inverse domain
• Database distributed over name servers
- Root server, TLD servers, authoritative servers
• Resolver performs (iterative) resolution on behalf of clients
• Name servers are responsible for zones
- Responsibilities are distributed through delegations
• Supports different kinds of queries
- A, AAAA, NS, PTR, MX, …
• BIND DNS software
- Zone file definitions

Lecture 13 IP Configuration 分IP地址

Automating IP Configuration--之前还提到了stateless和stateful的分别是HTTP和它的cookie！

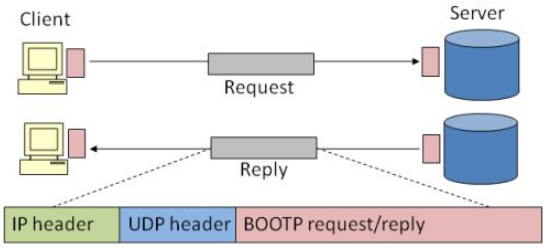
• **BOOTP** (Bootstrap Protocol)
- Static, stateful, client-server
• **DHCP** (Dynamic Host Configuration Protocol)
- Dynamic, stateful, client-server
• **SLAAC** (Stateless Address Autoconfiguration)
- Dynamic, stateless
- RFC 4862: IPv6 Stateless Address Autoconfiguration
• **Zeroconf**
- Autoconfiguration completely without servers?

△RARP—Reverse ARP



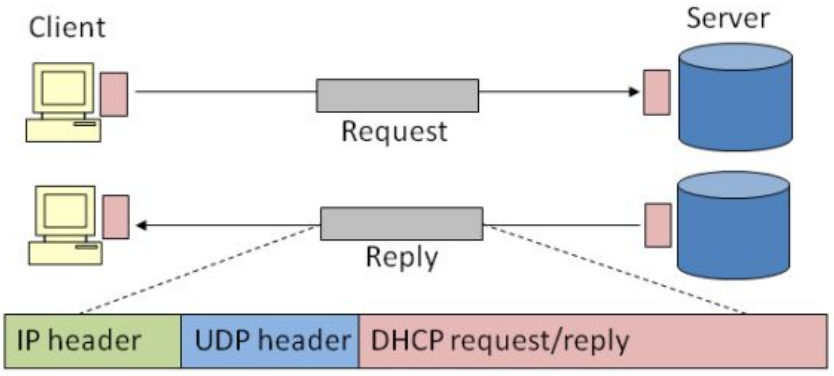
1. BOOTP—Bootstrap Protocol

- BOOTP (RFC 951) is a lot more powerful than RARP
- BOOTP sends requests/replies over UDP
  - Easy to write a user space server
  - Client does not need a full TCP/IP stack to run BOOTP
- BOOTP is not dynamic
  - static binding between MAC and IP addresses



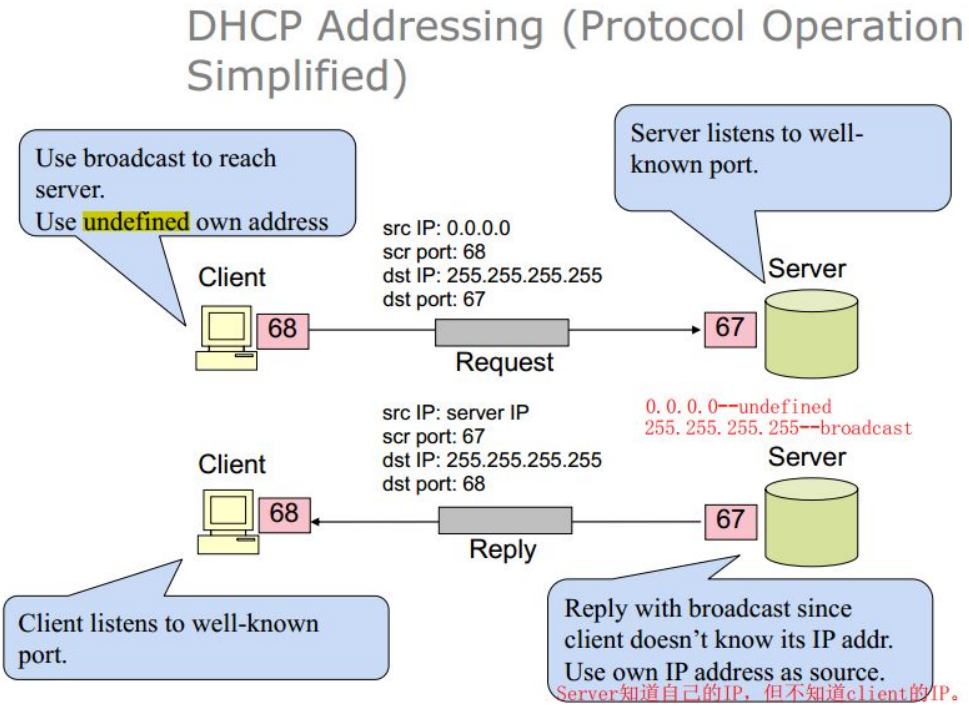
2. DHCP—Dynamic Host Configuration

- Main innovation: Dynamic configuration
  - Hosts can be assigned temporary IP addresses from a pool
    - Allows reuse of addresses
  - A subnet can thus support more hosts than available IP addresses (not all at the same time of course…)
    - Like KTH wireless LAN
- Also defines standard for additional information to clients
  - DNS server, time server, printers etc
- Backwards compatible with BOOTP
  - BOOTP clients can use DHCP server

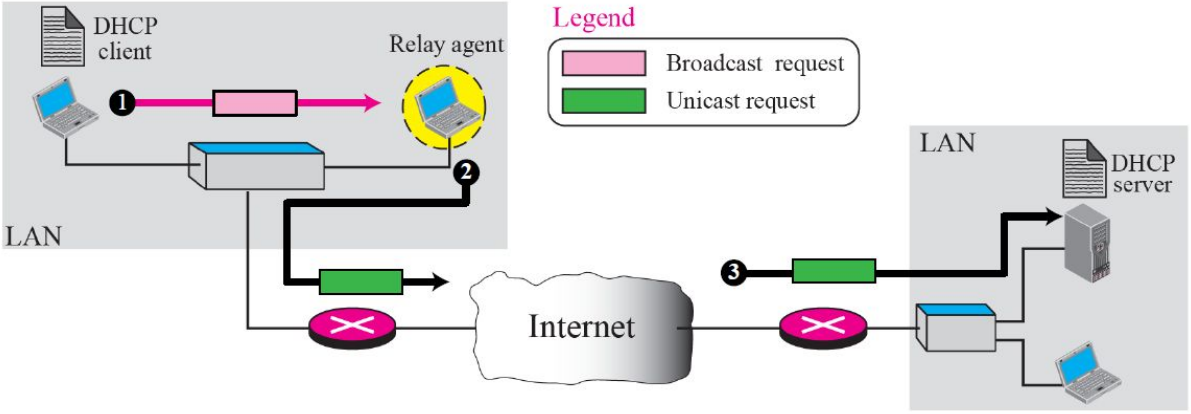


Client has neither its own IP address, nor the server's. How do we then address Request/ Reply?

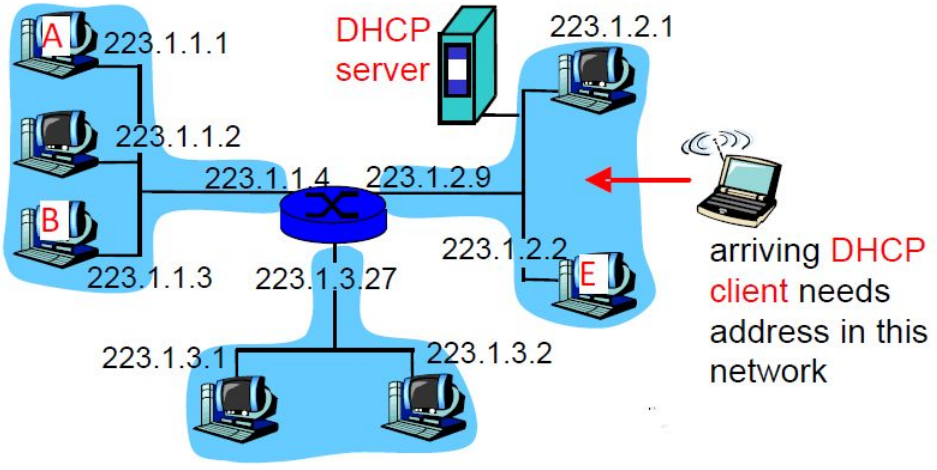
DHCP important:

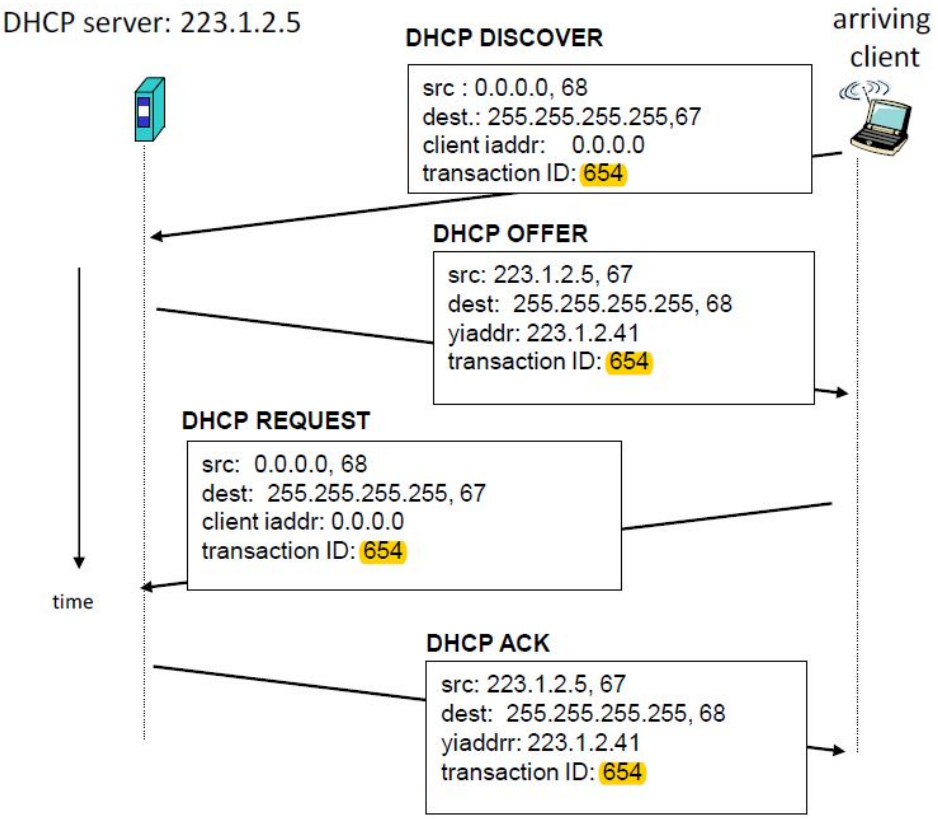


如果DHCP的client和server在不同的网络上，就需要relay agent：



DHCP Scenario（方案，分镜头）





Transaction ID should be the same value for all four messages.
Discover--**Offer**--**Request**--**ACK**

3. Host Configuration—SLAAC - - Stateless[18] Autoconfiguration
SLAAC vs DHCP
•Unfortunately, SLAACs contain only limited information
-**Prefixes** and **router address**
•Hosts often need other configuration information
-E.g., DNS server, Time server, Printer server
•For these, we still need DHCP servers
-Good news is that we can use SLAAC for setting up the IP address, and use stateless DHCP for everything else

4. IPv6 Autoconfiguration[19]—Plug and Play
•Idea: automatically discover parameters used to connect to the Internet
-Address, netmask, router, nameserver, ...
•Two scenarios: stateless and stateful
In IPv6 stateless autoconfiguration, the client can create an IP address based on its MAC address instead of requesting it from a DHCP server.

[18] Server keeps no state about hosts, only non-host state
[19] One of the interesting features of IPv6 addressing is the autoconfiguration of hosts. As we discussed in IPv4, the host and routers are originally configured manually by the network manager. However, the Dynamic Host Configuration Protocol, DHCP, can be used to allocate an IPv4 address to a host that joins the network. In IPv6, DHCP protocol can still be used to allocate an IPv6 address to a host, but a host can also configure itself.

①Advatage：A MAC-derived IPv6 address is a straight forward way to generate a unique IP address automatically and L3/L2 address translation can be done locally by the sender (no ARP needed).
②Problem： The MAC address reveals information about the interface card（L2，L3的地址, name），such as identity and vendor of the interface card, so that e.g. potential bugs could be exploited.
③Solution：IPv6 privacy extensions solve this problem by using a randomly assigned interface ID instead and this number can change over time (temporal address). **此时需要ARP**

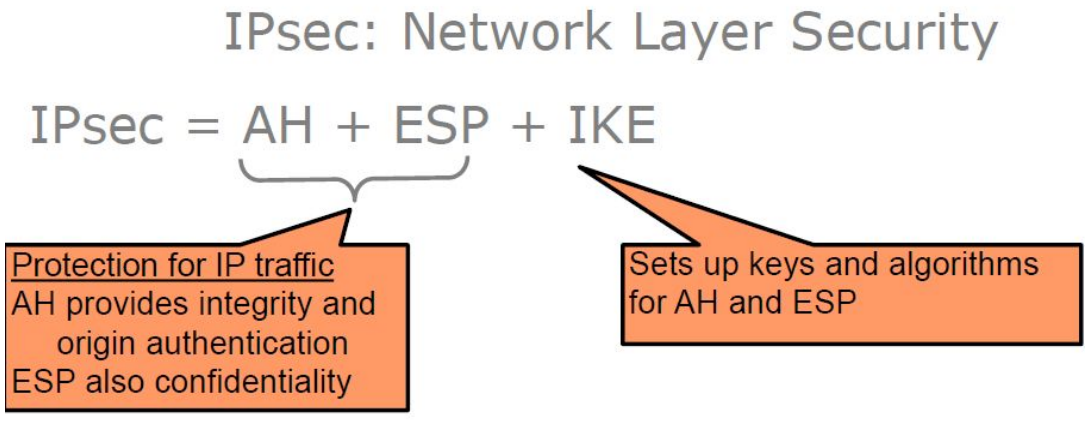
5. Stateful and Stateless Autoconfiguration

Stateless autoconf	Stateful autoconf
-Small networks -Nodes can start communicating directly	-Larger networks -Centralized management
Combination (Stateless DHCP) [DHCP本身是stateful的]	

Lecture 14 IP Security

1. Overview

•Authenticated Keying
–Internet Key Exchange (IKE)
•Data Encapsulation
–ESP: IP Encapsulating Security Payload (RFC 4303)
–AH: IP Authentication Header (RFC 4302)
•Security Architecture (RFC 4301)
–Tunnel/transport Mode
–Databases (Security Association, Policy, Peer Authorization)



•AH and ESP rely on an existing security association
–Idea: parties must share a set of secret keys and agree on each other's IP addresses and crypto algorithms
•Internet Key Exchange (IKE)
–Goal: establish security association for AH and ESP

–If IKE is broken, AH and ESP provide no protection!

▵IPsec Modes

•Transport mode

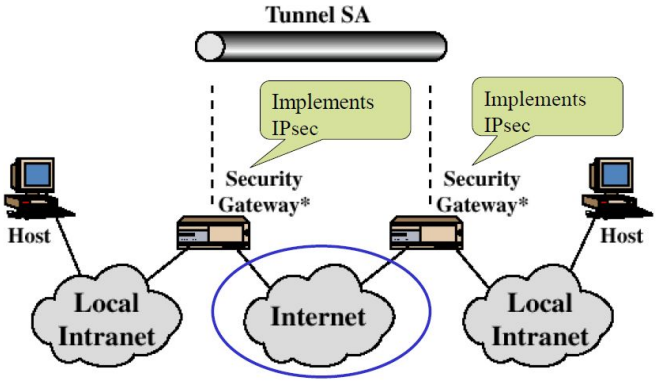
–Used to deliver services from host to host or from host to gateway

–Usually within the same network, but can also be end-to-end across networks

•Tunnel mode[20]

–Used to deliver services from gateway to gateway or from host to gateway

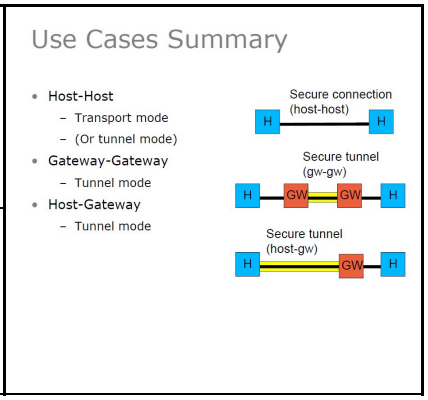
–Usually gateways owned by the same organization
(With an insecure network in the middle)



IPsec protects communication on the insecure part of the network.

两种mode对比：

Transport	secures packet payload and leaves IP header unchanged（只管payload） IP header (real dest) | IPsec header | TCP/UDP hdr + data
Tunnel	encapsulates both IP header and payload securely into IPsec packets（IPheader+Payload） IP header (gateway) | IPsec header | IP header (real dest) | TCP/UDP hdr + data



▵Security Association (SA)

•One-way sender-recipient relationship

–Manually configured or negotiated through IKE

•SA determines how packets are processed

–Cryptographic algorithms, keys, AH/ESP, lifetimes, sequence numbers, mode (transport or tunnel)

•SA is uniquely identified by **{SPI, dst IP addr, flag}**

–SPI: Security Parameter Index

•Chosen by destination (unless traffic is multicast...)

–Flag: ESP or AH

–Each IPsec implementation keeps a database of SAs

–SPI is sent with **packet**, tells recipient which SA to use

[20] tunnel mode's typical application: virtual private network (**VPN**，通常由**ESP**实现）

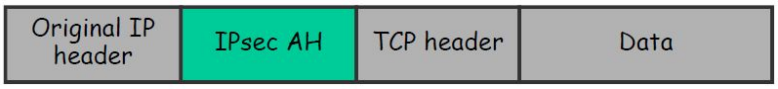
2. Encapsulation Formats

1) AH

–Authentication Header

–Provides integrity

Only in transport mode:

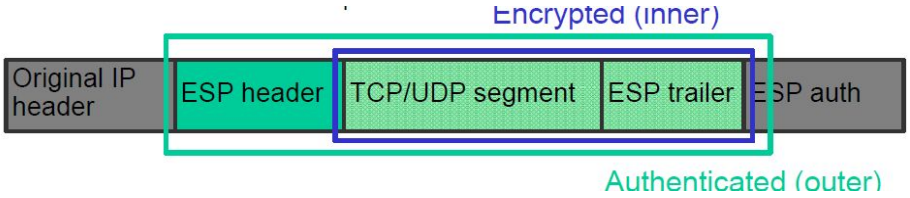


2) ESP

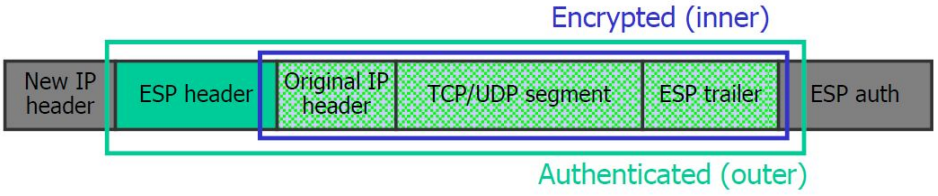
–Encapsulating Security Payload

–Provides integrity and/or privacy

Can work in transport…(original IP header没有被封装起来)



…or tunnel mode (problem with NAT)



！ ！ ！Tunnel mode can be problematic together with NAT

•If we set up a tunnel between our host and a public gateway, it won't work:

–Our private addresses will be in the original IP header

•It is OK to set up a tunnel between our host and a private intranet:

–Private intranet addresses will be in the original IP header

–New IP header will contain our home private address, which will be translated by the NAT

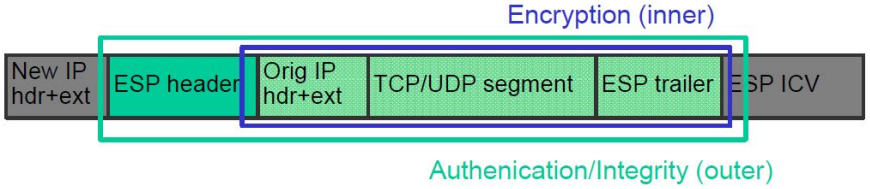
3. IPsec and IPv6

IPsec is a mandatory component for IPv6.

Extension headers are used for IPsec.

IPsec Tunnel Mode in IPv6:

- IPv6 IPsec is implemented using
  - Authentication extension header
  - ESP extension header



4. IKE

•Internet Key Exchange—setting up the SAs for IPsec (ESP and AH SA's)

•Use IKE protocol to do mutual authentication and to create a session key

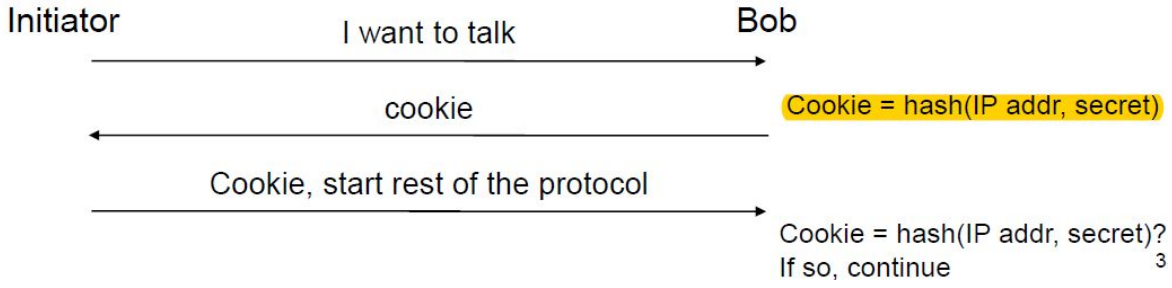
–Use Diffie-Hellman to derive shared symmetric key

▵Diffie-Hellman

•For IKE to use Diffie-Hellman we need to add

–**Cookies** for protection against **denial-of-service attacks**

- Solution:
  - When Bob receives connection initiation from IP addr S
    - Send unpredictable number (cookie) to S—should be stateless!
    - Do nothing until same cookie is received from S
  - Assures that initiator can receive packets sent to S



The cookies should be stateless，so “Bob” doesn’t have to keep track of all cookies he sent.

–**Nonces** to ensure against **replay attacks**

▵IKE Phases

•Phase 1

–do mutual authentication and establish IKE session keys

–Sets up the “main” SA (or IKE SA)

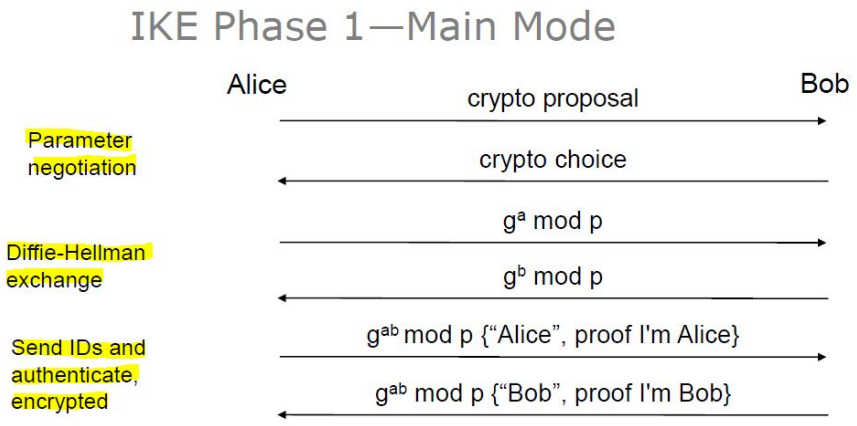
•Phase 2

–Set up one or more IPsec SAs (child SAs) between the nodes using the **keys derived in phase 1**

•Why two phases?

–Mutual authentication is expensive

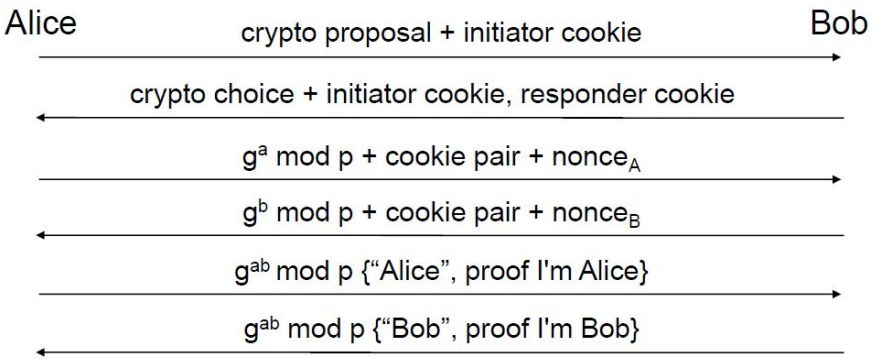
–If multiple SAs are needed or if SA parameters need to be changed, this can be done without repeating mutual authentication



- Proof of identity different for different key types
  - Pre-shared secret, private encryption or signature key,...
- Proof is a hash of
  - key, Diffie-Hellman values, nonces, crypto choices, cookies

加入cookie和nounces之后：

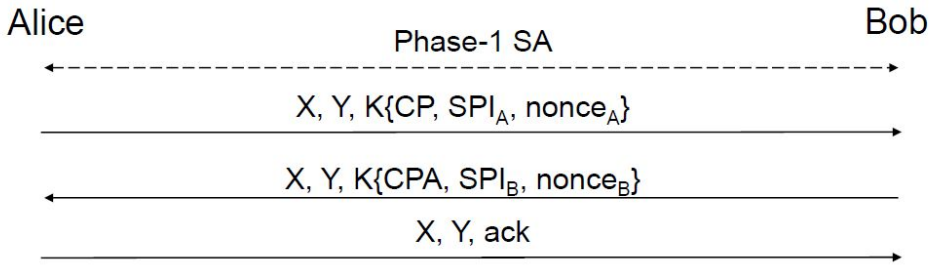
- More details: cookies and nonces



Recommended method for creating the cookie:

- Fast hash (e.g., MD5) over
  - IP src/dst addr, UDP src/dst port, locally generated secret value

IKE Phase 2, Setting up IPsec SAs



- X is a pair of cookies from phase 1
- Y is a 32-bit number
  - ID to distinguish between multiple phase 2 sessions
- The rest is encrypted using SKEYID_e and authenticated using SKEYID_a
  - This part is simplified—more info can be exchanged

Lecture 15 IP Gateways

这一章首先介绍了什么事gateway--

A machine that sits between two interconnected networks and relays traffic between them.

Traffic cannot flow between the two networks without the assistance of the gateway.

Conclusion: A routeris a **network layer** gateway

–But we can have other types of gateways, both at the network layer and elsewhere

都有什么特殊功能呢

1. Connecting networks with incompatible (不兼容的) address systems，比如：

•IPv4 and IPv6----IPv4/IPv6 Gateways(不是重点)

•Two IPv4 networks with independent address domains-----NAT

2. Restricting what traffic flows between two networks-----firewall

•Protective purposes

3. Redirecting traffic, possibly tunneling it

•Mobility, VPNs, IPsectunnels etc

1. Firewall

Isolates organization's internal network from larger Internet, allowing some packets to pass and blocking others.

▵**Firewall Locations in the Network**

•Between internal LAN and external network

•At the gateways of sensitive subnetworkswithin the organizational LAN

-Payroll's network must be protected separately within the corporate network

•On end-user machines

-"Personal firewall"

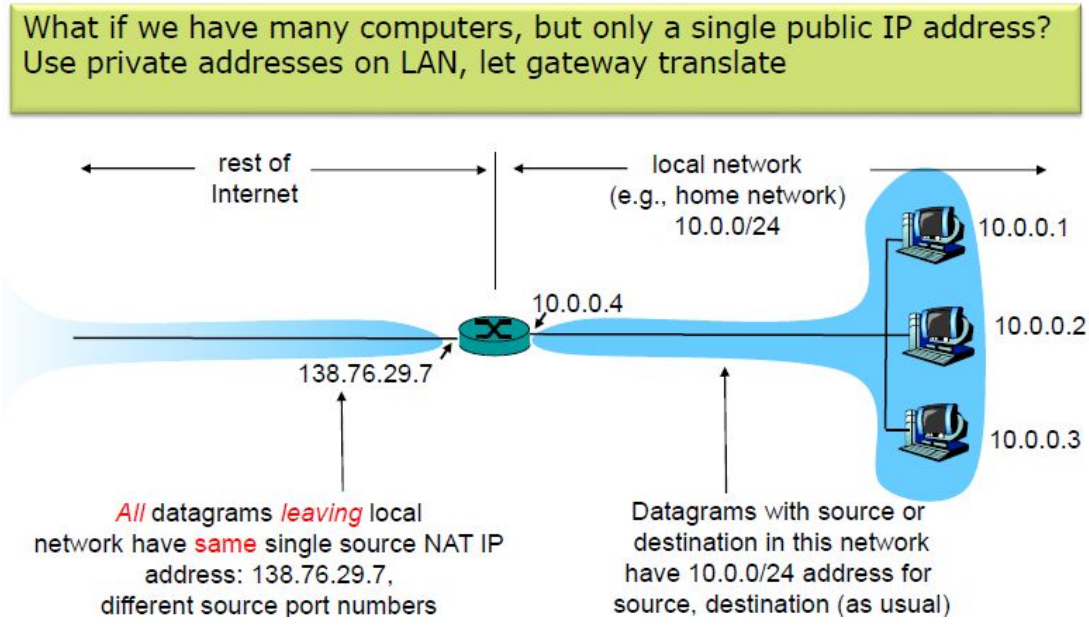
-Microsoft's Internet Connection Firewall (ICF)

▵**Firewall types**

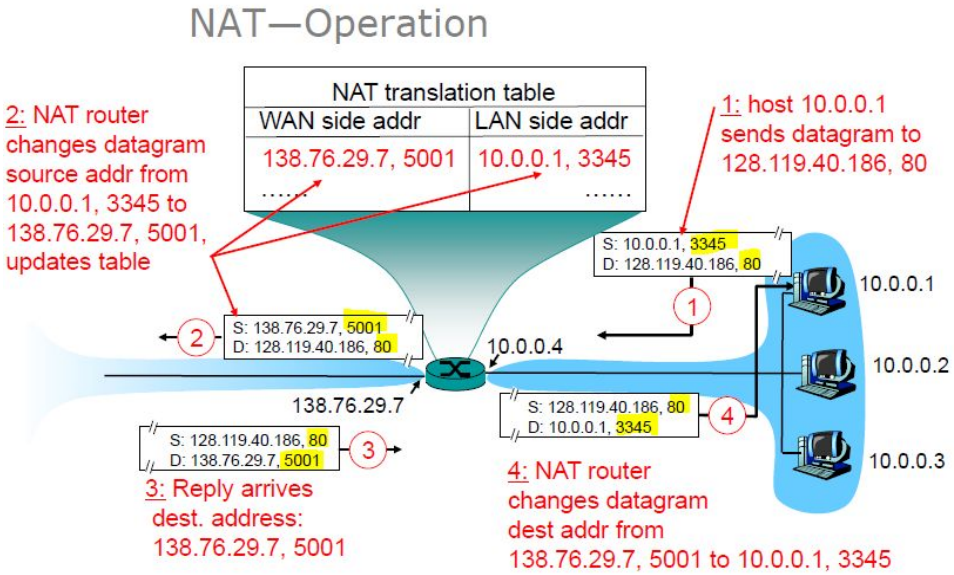
Packet filter: internal network connected to Internet via *router firewall*	Application level gateway: splices and relays two application-specific connections
Two default policies: •Default = discard -which is not explicitly permitted is prohibited •Default = forward -which is not explicitly prohibited is permitted •Default = discard is more conservative	
对比： •Packet filter can do its job without requiring software changes in communicating nodes -Allowed conversations proceed normally (in most cases) •An application level gateway is visible to the users -Need to connect to the gateway	

•Application level gateway can be more powerful than packet filters—e.g., look at data inside email messages
-Gateway is application-aware

2. NAT - Network Address Translation



右边的这些local network用138.76.29.7的不同端口表示。
10.0.0.1到10.0.0.4是可以在其他local network中重复利用的，而138.76.29.7是唯一的。
例子：Assume that **host 10.0.0.1** on a **private network (10.0.0.0/24)** sends an HTTP request through its NAT box to **a web server on address 128.119.40.186** and that this web server answers with an HTTP response back to the host.



1) port 80是web server的默认HTTP端口，固定的; 10.0.0.1的port3345不是固定的
2) 最终138.76.29.7, port 5001被分配给10.0.0.2。（5001不是固定的，只是这么分配而已，可以给下面两个hosts分配5002，5003）
3) 两个方向上每次都是138和10开头的地址替换，webserver的地址一直为Source / Dest.

总图

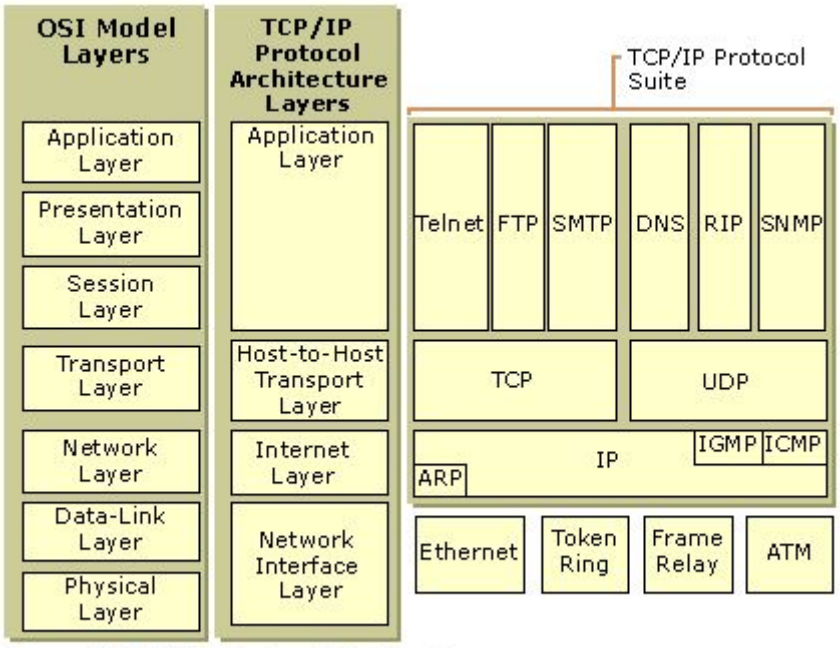


Figure 1.1 TCP/IP Protocol Architecture