1.1.1

S1：

|  |  |
| --- | --- |
| Destination | Port |
| A | 1 |
| B | 2 |
| Default | 3 |

S2：

|  |  |
| --- | --- |
| Destination | Port |
| A | 1 |
| B | 1 |
| C | 3 |
| D | 3 |
| Default | 2 |

S3：

|  |  |
| --- | --- |
| Destination | Port |
| C | 2 |
| D | 3 |
| Default | 1 |

S4：

|  |  |
| --- | --- |
| Destination | Port |
| D | 2 |
| Default | 1 |

1.2.2

1. -D：S11--S21--S31
2. -B：S12
3. -D：S13--S22--S32

S11:S1交换机的1号端口

1.3.3

透明网桥的工作原理：

1）自学习和转发帧

下面是网桥的自学习和转发帧的一般步骤。

网桥收到一帧后先进行自学习。查找转发表中与收到帧的源地址有无相匹配的项目。如果没有，就在转发表中增加一个项目。如果有，则把原有的项目进行更新。

转发帧。查找转发表中与收到帧的目的地址有无相匹配的项目。如果没有，则通过所有其他接口进行转发。如果有，则按转发表中给出的接口进行转发。但应注意，若转发表中给出的接口就是该帧进入网桥的接口，则应丢弃这个帧。

1. 逆向学习法

透明网桥采用的算法是逆向学习法(backward learning)。网桥按混杂的方式工作，故它能看见所连接的任一LAN上传送的帧。查看源地址即可知道在哪个LAN上可访问哪台机器，于是在散列表中添上一项。

　　当计算机和网桥加电、断电或迁移时，网络的拓扑结构会随之改变。为了处理动态拓扑问题，每当增加散列表项时，均在该项中注明帧的到达时间。每当目的地已在表中的帧到达时，将以当前时间更新该项。这样，从表中每项的时间即可知道该机器最后帧到来的时间。网桥中有一个进程定期地扫描散列表，清除时间早于当前时间若干分钟的全部表项。于是，如果从LAN上取下一台计算机，并在别处重新连到LAN上的话，那么在几分钟内，它即可重新开始正常工作而无须人工干预。这个算法同时也意味着，如果机器在几分钟内无动作，那么发给它的帧将不得不散发，一直到它自己发送出一帧为止。

　　到达帧的路由选择过程取决于发送的LAN(源LAN)和目的地所在的LAN(目的LAN)，如下所示：

　　1、如果源LAN和目的LAN相同，则丢弃该帧。

　　2、如果源LAN和目的LAN不同，则转发该帧。

　　3、如果目的LAN未知，则进行扩散。

为了提高可靠性，有人在LAN之间设置了并行的两个或多个网桥，但是，这种配置引起了另外一些问题，因为在拓扑结构中产生了回路，可能引发无限循环。

3）生成树算法

透明网桥还使用了一个生成树（spanning tree）算法，即互连在一起的网桥在进行彼此通信后，就能找出原来的网络拓扑的一个子集。在这个子集里，整个连通的网络中不存在回路，即在任何两个站之间只有一条路径。

为了得能够反映网络拓扑发生变化时的生成树，在生成树上的根网桥每隔一段时间还要对生成树的拓扑进行更新。

透明网桥的路径选择算法归纳

（1）若目的局域网和源局域网一样，则网桥将该帧删除。

（2）若源局域网和目的局域网是不同的网，则将该帧转发到目的局域网。

（3）若目的局域网不知道，则采用扩散法处理。

遇到的问题：不能最佳地利用网络带宽

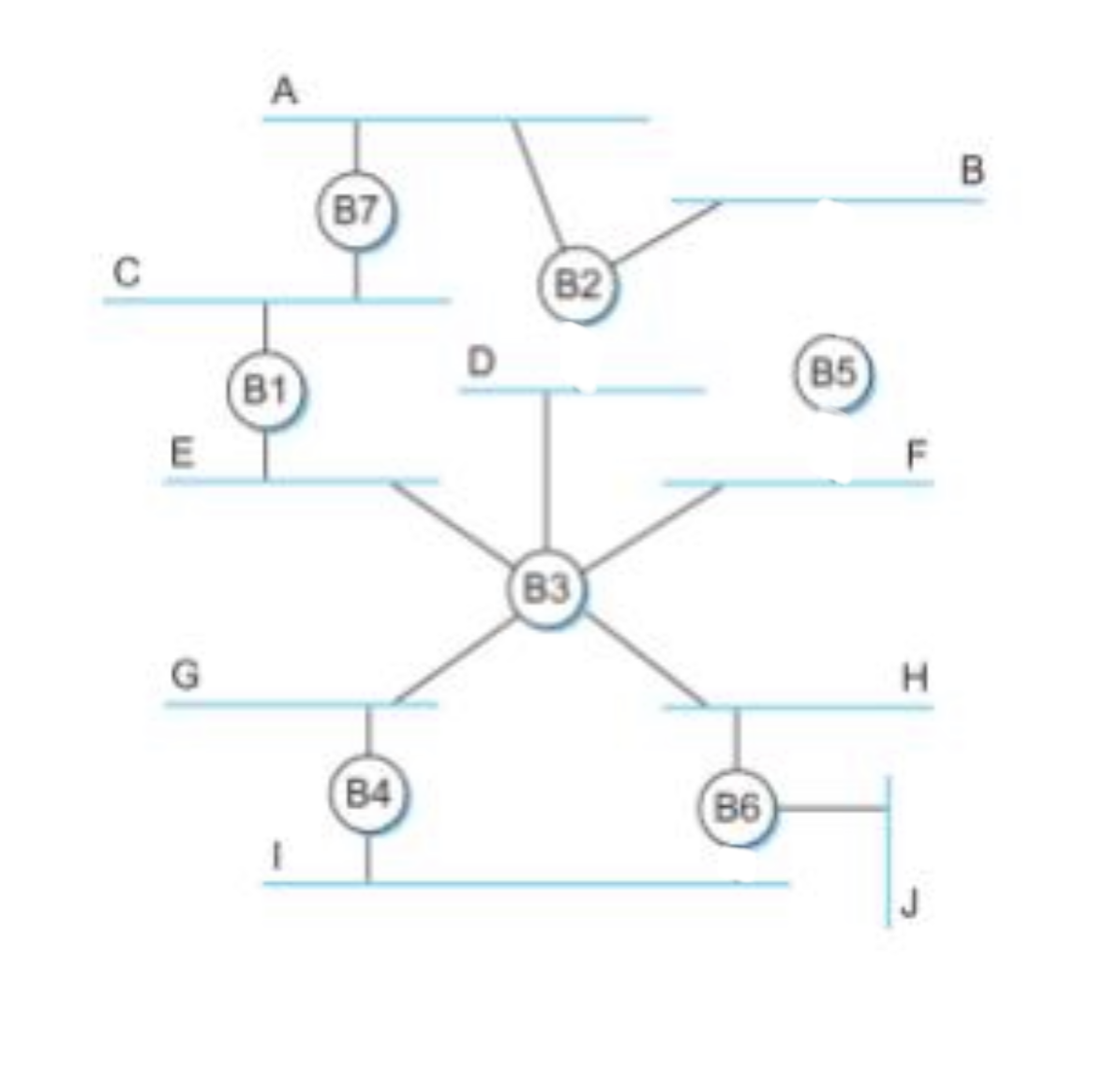
1.3.4

1. 所有网桥都能学习到X的位置，Y的网络接口能收到这个分组。
2. 当Z向A发送数据包时，所有的网桥都已经知道X的位置，所以网桥在B3-B2-B1-X的链路上传送数据包，因为数据包经过了所有的网桥，所以所有网桥都能知道Z的位置；Y的网络接口不能接到这个数据包，因为B2网桥只将数据包发送给B1。
3. 当Y发送给X时，B2将把数据包发送给B1,让后B1发送给X，因此网桥B2和B1知道Y的位置，Z的网络接口不能收到这个分组。
4. 当W发送给Y时，B3不知道Y的位置，因此在所有的链路上发送，Z的网络接口可以收到这个包，当数据包到达B2时，它只发送给Y。因此只有B2和B3知道W的位置。

1.3.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 发送的帧 | 左网桥的转发表 | | 右网桥的转发表 | | 左网桥的处理  转发？丢弃？登记？ |
| 地址 | 接口 | 地址 | 接口 |
| A--E | A | 1 | A | 1 | 转发 |
| C--B | C | 2 | C | 1 | 转发 |
| D--C | D | 2 | D | 2 | 丢弃 |
| B--A | B | 2 | B | 1 | 转发 |

1.3.6



1.4.8

（100/N）Mbps

1.4.9

1.5.1

|  |  |  |  |
| --- | --- | --- | --- |
|  | 数据字段长度 | 片偏移字段 | MF标志 |
| 片1 | 1480 | 0 | 1 |
| 片2 | 1480 | 185 | 1 |
| 片3 | 1020 | 370 | 0 |

固定首部占20字节，总数据部分长度为4000-20=3980，因为最大传送长度为1500，减去固定首部，长度为1480

片数据长度：片1为1480，片2为1480，片3为3980-1480-1480=1020

片偏移字段：IP数据包里的OFFSET应该是分组偏移量，

片1为0；

因为1480/8=185，所以片2为185；

因为（1480+1480）/8=370，所以片3为370

MF：后面还有片就是1，没有片就是0，因此片1和片2都为1，片3为0

1.5.2

（a）0.1×0.1=0.01

（b）单个分段两个批次传输都丢失的概率为0.01×0.01=0.0001，整个数据包丢弃事件发生的概率是每个分段两个批次传输都丢失的概率之和，因此为0.0001×10=0.001

（c）当一个分段丢失重传的时候，保持该分段的Ident字段不变。当重传延时小于组装延时，（b）中所述情况就可能发生，每一个接受到的数据包可能来自不同的批次。

1.5.3

192.12.6.0/23 => |1100 0000. 0000 1100. 0000011|0. 0000 0000

A: 192.12.0000 0110.|0000 0000/24 ==192.12.6.0/24 可用主机数目 2^(32-24)=256 >157

B: 192.12.0000 0111.0|000 0000/25 ==192.16.7.0/25 可用主机数目 2^(32-25)=128 > 128

C: 192.12.0000 0111.10|00 0000/26 == 192.12.7.128/26 可用主机数目2^(32-26)=64 > 51

D: 192.12.0000 0111.110|0 0000/27 == 192.12.7.192/27 可用主机数目2^(32-27)=32 > 24

D: 192.12.0000 0111. 1110 |0000/28 == 192.12.7.224/28 可用主机数目2^(32-27)=32 > 16

1.5.4

(a)

(212.1.1.0和212.1.1.255)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | B | C | D |
| IP（212.1.1） | 1~127 | 128~191 | 192~224 | 224~254 |
| 子网掩码 | 255.255.255.128 | 255.255.255.192 | 255.255.255.224 | 255.255.255.224 |

（b）同（a）

1.5.5

两种方法: IP 地址和 MAC 地址。

区别：

1. 长度不同。IPv4 地址长度为 32 位；MAC 地址长度为 48 位。

2. 对应的协议不同。IP 地址对应 IP 协议；MAC 地址对应 Ethernet 协议。

3. 寻址协议层不同。IP 地址在网络层；MAC 地址在数据链路层。

4. 分配依据不同。IP 地址分配基于网络拓扑；MAC 地址分配基于制造商。

5. 结构不同。IP 地址有层次结构，可划分子网；MAC 地址不具备层次结构。

联系：

使用统一的、结构化的 IP 地址指定主机之间、网络之间的路由关系，由 ARP 解析 到对应的 MAC 地址，完成底层的数据转发。

1.5.6

1. 如果在第一个突发的，目的地分组为D的分组到达IP层之后且第一次ARP请求发出但未得到响应之前，有多个目的地为D的分组到达，这样会发出多个相同的ARP请求，这样不仅浪费了带宽，而且因为它们是广播出去的，它们会打断每个主机和网桥。
2. 我们应该保存一个当前时间之前重复发生的ARP请求的列表，当要发出一个请求之前，我们应该先进行查表，如果这个请求不在列表中，我们再发送请求。
3. 这可能导致在新的连接刚刚建立起来时会发生频繁大量的丢包现象。

1.6.7

1.6.8

（a）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Distance | | | | | |
| A | B | C | D | E | F |
| A | 0 | **∞** | 3 | 8 | **∞** | **∞** |
| B | **∞** | 0 | **∞** | **∞** | 2 | **∞** |
| C | 3 | **∞** | 0 | **∞** | 1 | 6 |
| D | 8 | **∞** | **∞** | 0 | 2 | **∞** |
| E | **∞** | 2 | 1 | 2 | 0 | **∞** |
| F | **∞** | **∞** | 6 | **∞** | **∞** | 0 |

（b）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Distance | | | | | |
| A | B | C | D | E | F |
| A | 0 | **∞** | 3 | 8 | 4 | 9 |
| B | **∞** | 0 | 3 | 4 | 2 | **∞** |
| C | 3 | 3 | 0 | 3 | 1 | 6 |
| D | 8 | 4 | 3 | 0 | 2 | **∞** |
| E | 4 | 2 | 1 | 2 | 0 | 7 |
| F | 9 | **∞** | 6 | **∞** | 7 | 0 |

（c）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Distance | | | | | |
| A | B | C | D | E | F |
| A | 0 | 6 | 3 | 6 | 4 | 9 |
| B | 6 | 0 | 3 | 4 | 2 | 9 |
| C | 3 | 3 | 0 | 3 | 1 | 6 |
| D | 6 | 4 | 3 | 0 | 2 | 9 |
| E | 4 | 2 | 1 | 2 | 0 | 7 |
| F | 9 | 9 | 6 | 9 | 7 | 0 |

1.6.9

|  |  |  |
| --- | --- | --- |
| 步骤 | 路径 | 尝试 |
| 1 | (A,0,-) |  |
| 2 | (A,0,-) | (D,2,D)(B,5,B) |
| 3 | (A,0,-)(D,2,D) | (B,4,D)(E,7,D) |
| 4 | (A,0,-)(D,2,D)(B,4,D) | (E,6,D)(C,8,D) |
| 5 | (A,0,-)(D,2,D)(B,4,D)(E,6,D) | (C,7,D) |
| 7 | (A,0,-)(D,2,D)(B,4,D)(E,6,D)(C,7,D) |  |

1.7.10

1. B
2. A
3. E
4. F
5. C
6. D

1.7.11