第五章 网络层（1）

袁华：hyuan@scut.edu.cn
华南理工大学计算机科学与工程学院
广东省计算机网络重点实验室（CCNL）

第五章的主要目标

- 分组如何从源机到达目的机？
- 怎样找到一根通路？（Path）
  - 路由选择协议，路由器（Router）
  - 被路由协议： IPv4、IPv6
  - 其它，如ARP、ICMP、CIDR等

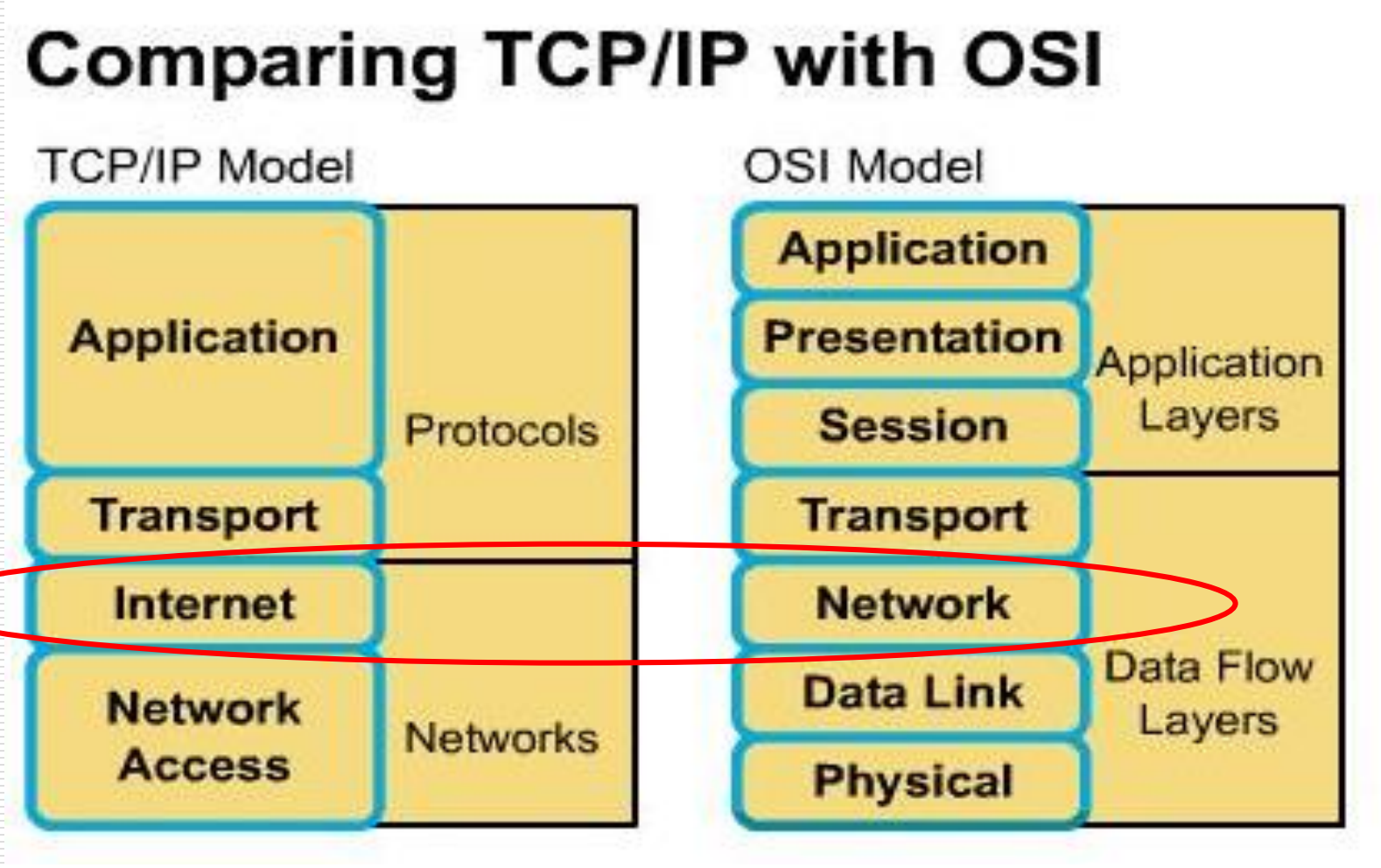
第一部分的主要内容（5.1~5.2 P274~302）

- 网络层概述
- 路由选择协议概述（**Routing algorithm**）
- 学习 **Dijkstra** 算法
- 距离矢量（**DV**）路由选择算法：**Rip**
- 链路状态（**LS**）路由选择算法 ： **OSPF**
- 多级路由、广播路由、移动路由、对等路由、**P2P**等等

本节的主要内容

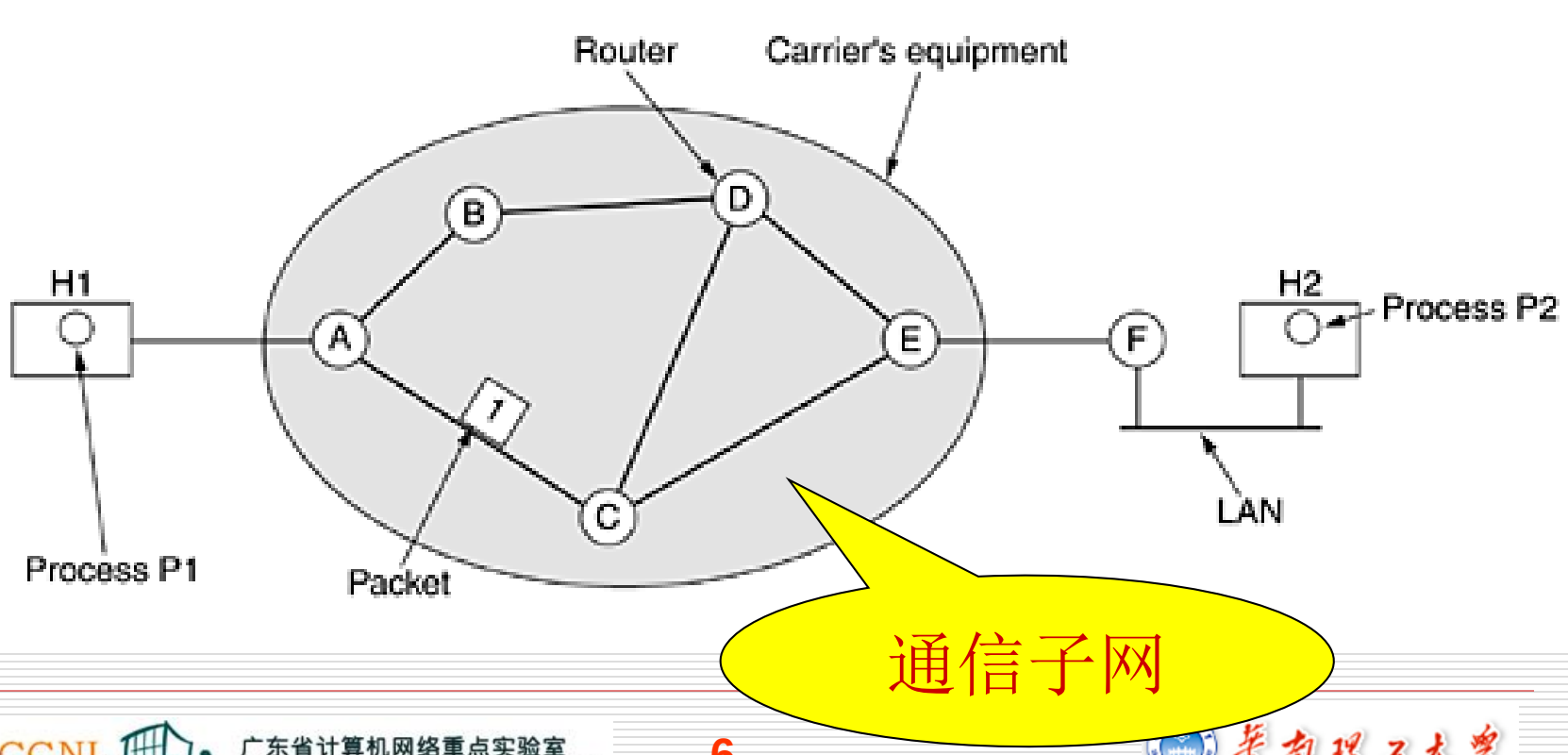
- 网络层的主要功能
- 路由选择算法
  - 静态路由选择算法
    - Dijkstra
    - flooding
  - 动态路由选择算法
    - 距离矢量路由（DV）
    - 链路状态路由（LS）
- 掌握 Dijkstra 算法

网络层的位置


Comparing TCP/IP with OSI
TCP/IP Model
Application
Transport
Internet
Network Access
Protocols
Networks
OSI Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
Application Layers
Data Flow Layers

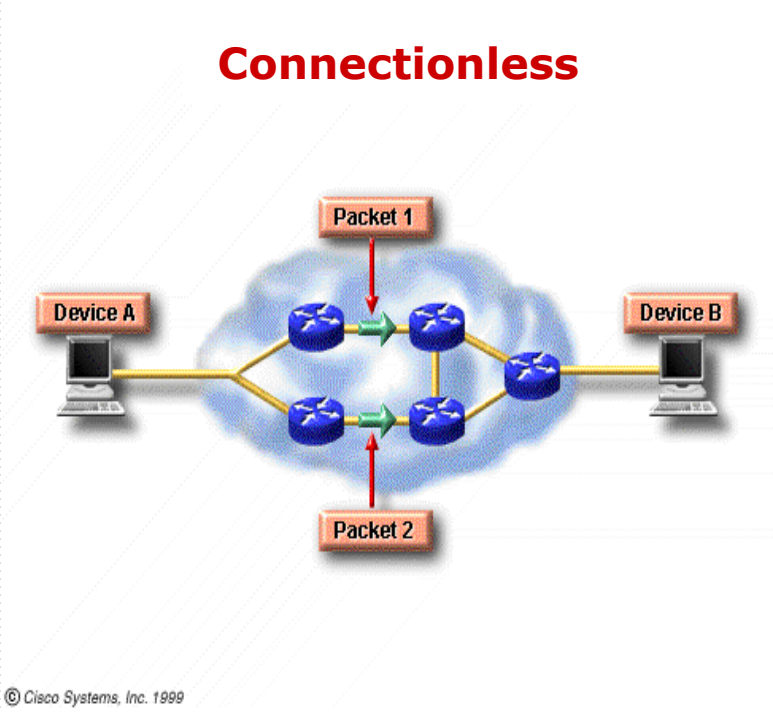
网络层的主要功能

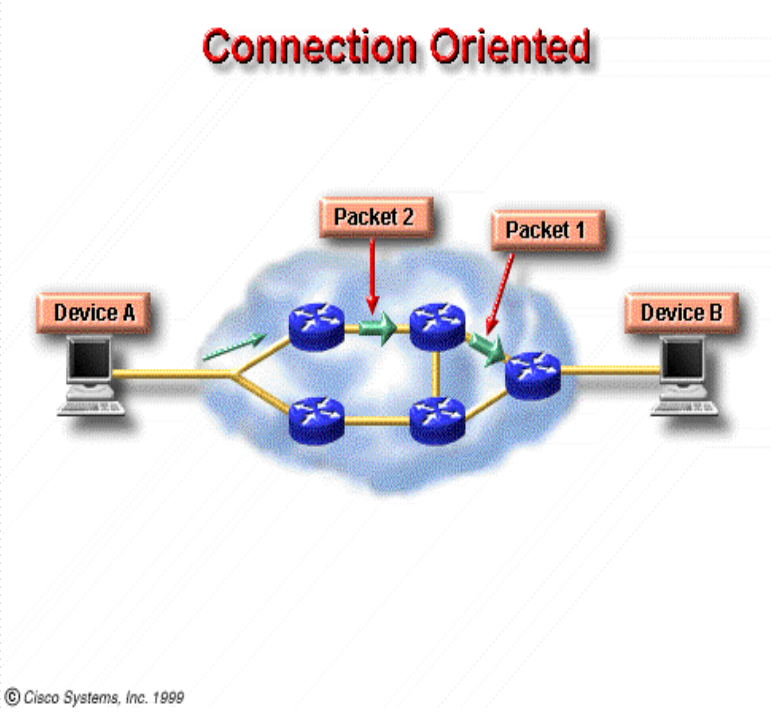
- 主要功能就是：将分组从源机一路送到目的机 **P274**



网络层提供的服务 P275

- 面向连接的服务：X.25, ATM
- 无连接的服务：IP

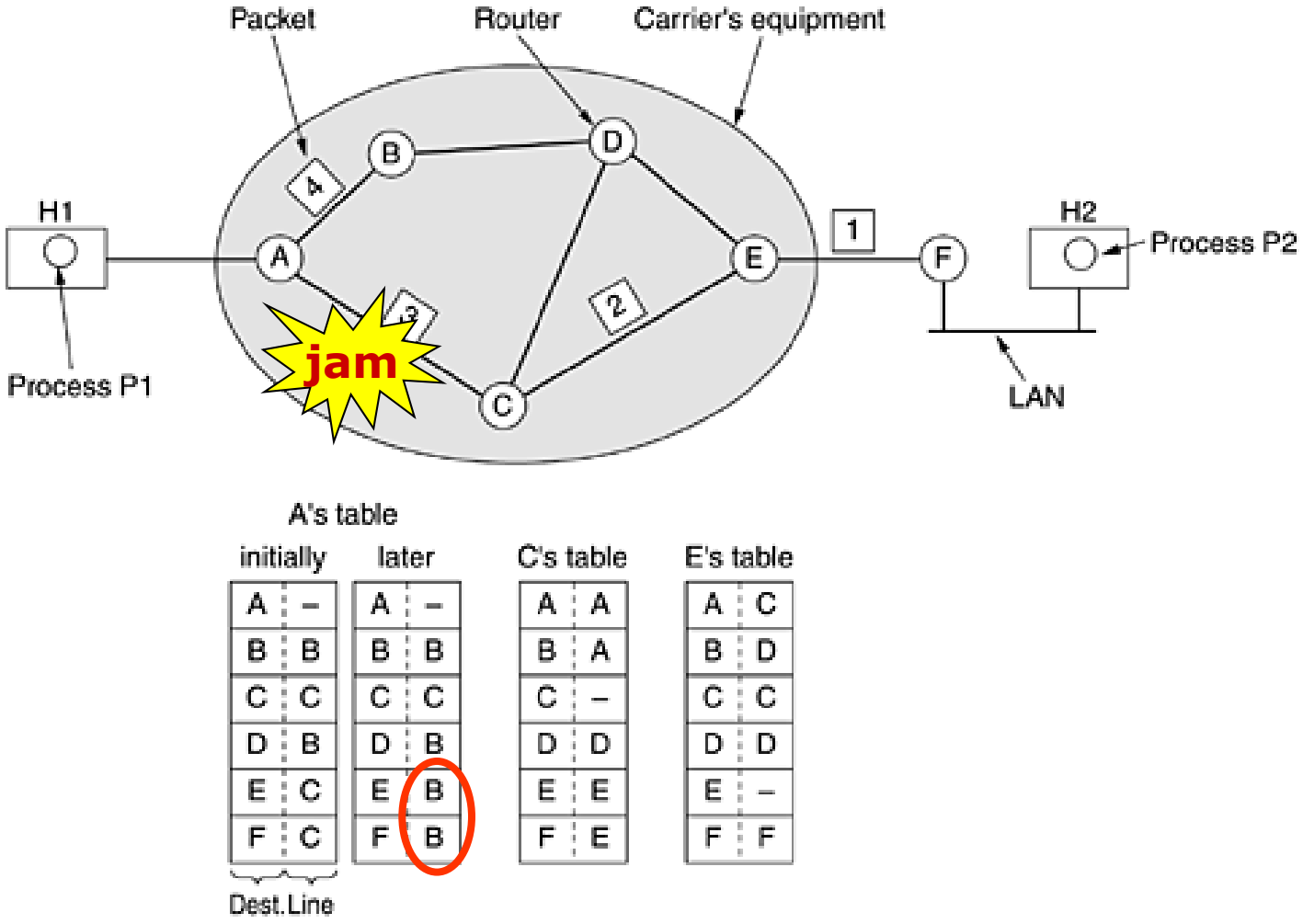




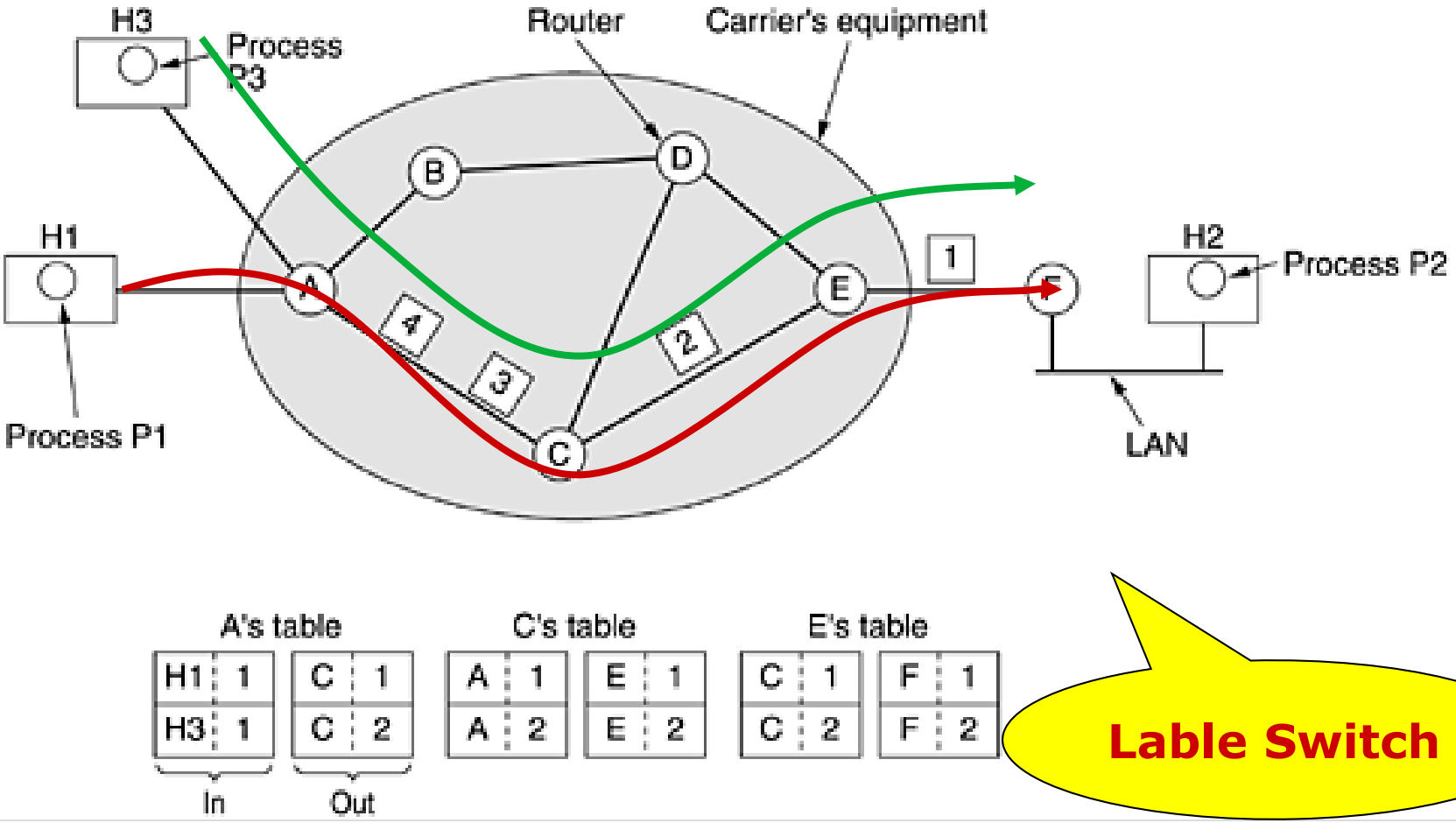
对应的通信子网的结构P276~278

- 虚电路子网（Virtual-circuit subnet）
  - 在连接建立的时候选路（Select a path）
  - 每个分组携带一个连接号（connection-number）
  - 当通信完成后，连接拆除
- 数据报子网（Datagram subnet）
  - 每个数据报携带目的地址
  - 每个报文独立寻径

无连接的服务—数据报子网P276



面向连接的服务-虚电路子网P277~278



两种通信子网的比较

- 虚电路子网
  - 通过路径选择后建立连接
  - 到终点后毋需重新排序
  - 每个分组不需带目的地址，但带虚电路号（较短）
  - 主机工作量少，差错检查、流量控制对用户透明。
- 数据报子网
  - 子网工作简单，通信费用低。
  - 每个分组分别选择最佳路径，健壮性较好
  - 到终点后需重新排序
  - 差错控制和排序工作由协议高层（主机）完成
  - 每个分组必须带目的地址，路径选择灵活。

比较表P278

比较项目	数据报子网（无连接服务）	虚电路子网（面向连接服务）
建立电路	不需要	要求
地址信息	每个分组含完整的SA和DA	每个VC包含一个很短的VC号码
状态信息	路由器不保留任何连接状态信息	每个VC都要求路由器建立表项
路由	每个分组独立选择路由	每个分组沿建立VC时确定的路由
路由器失效影响	没有，只有系统崩溃时丢失分组	所有经过失效R的VC都终止
服务质量	很难实现	总资源（带宽、缓存）足够的情况下，采用提前给每个VC分配资源的方法，很容易实现
拥塞控制		

问题

- 虚电路子网是否不需要进行路径选择？（path-select 、routing algorithm)?

路由表是如何建立起来的?

- 静态路由
  - 由管理员手工配置的：**ip route**
- 动态路由
  - 路由选择算法
    - 距离矢量路由选择（**D-V**）
    - 链路状态路由选择（**L-S**）

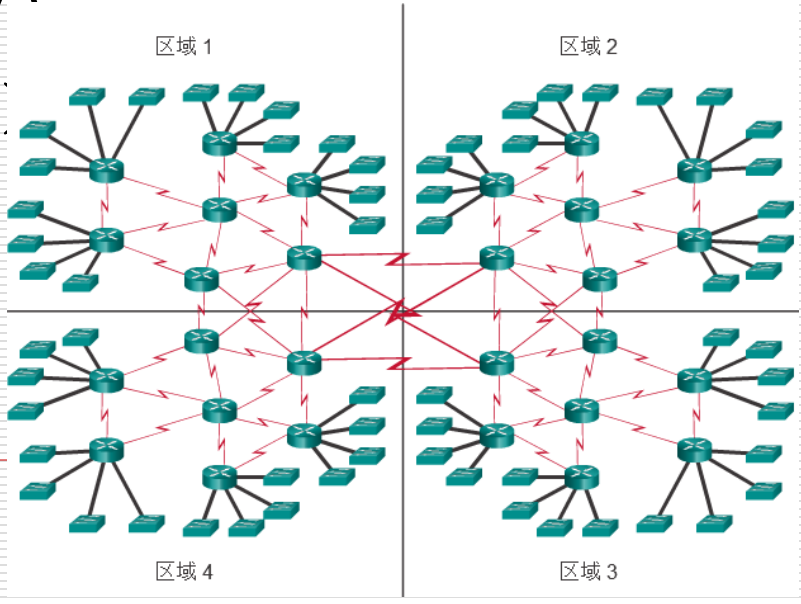
什么时候使用静态/动态路由？

- 静态路由：管理员手工配置的路由
  - 适合小型的、静态的网络，开销小
  - 缺省路由：默认路径，找不到路的时候可以从这里通过
    - 避免错误丢包
    - 缩减路由表的规模
    - 减少路由器的运行负担
- 主机路由表：**route print**

网络目标	网络掩码	网关	接口	跃点数
0.0.0.0	0.0.0.0	192.168.1.1	192.168.1.105	25
127.0.0.0	255.0.0.0	在链路上	127.0.0.1	306
127.0.0.1	255.255.255.255	在链路上	127.0.0.1	306

动态路由选择协议分类

- 动态路由：由路由选择协议动态地建立、更新和维护的路由
  - 适合大型的、经常变动的网络，需要维护开销
  - 减少了网络管理员的负担
- 路由选择协议（Routing protocol）
  - 距离矢量路由选择协议（DV）
  - 链路状态路由选择协议（LS）
  - 混合路由选择协议

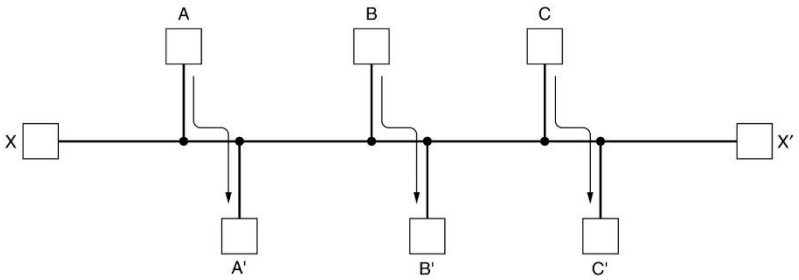


路由选择算法P279

- 什么是路由选择算法?P279
- 路由选择算法必须考虑这些因素:
  - 正确性、简单性(KISS)、健壮性（robustness）、稳定性、公平性和最优性 (contradictory、trade-off)
- 路由选择算法的分类P279~280
  - 静态算法（not self-adaptive）
  - 动态算法（ self-adaptive ）

路由算法的设计目标

- 正确性：全网路由信息一致且正确
- 简单性：算法简单有效
- 健壮性：适应各种网络环境和变化
- 稳定性：算法快速收敛
- 公平性和最优性：寻找两者的平衡点

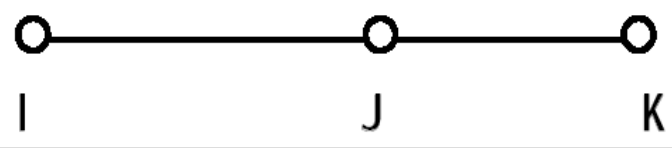


选择路由算法的度量参数（Metric）P280

- 别名：cost，量度、代价、开销、成本
- 常用的度量参数
  - 路径长度：由网络管理员定义每条网络链路的代价(cost)，从源到宿的代价总和为路径长度，hop (跳数)
  - 可靠性：链路数据传输的可靠性（误码率）
  - 延迟：数据包从源到宿需要花费的传输时间
  - 带宽：链路的最大传输能力以及网络流量
  - 负载：网络资源（例如路由器的CPU）
  - 通信代价：占用通信线路的费用

最优化原理（Optimization principle）P281

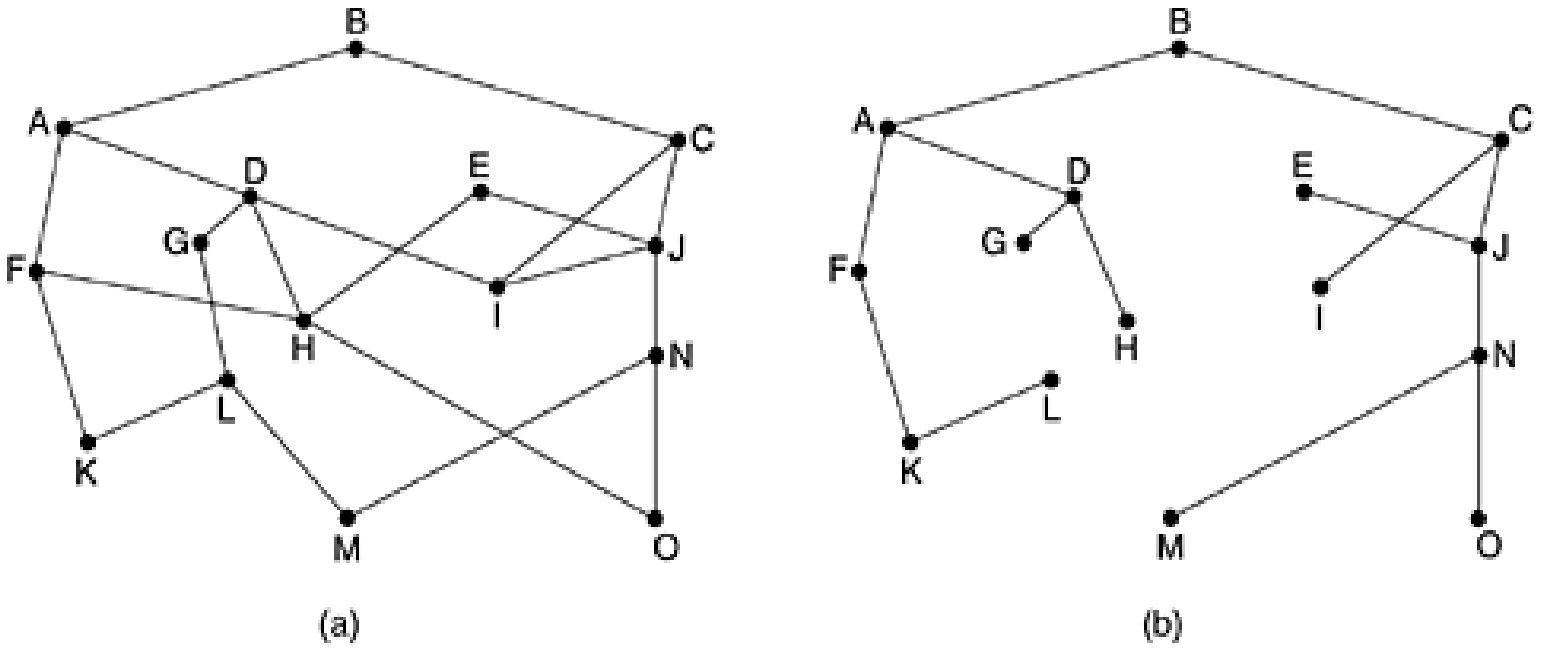
- 如果一个路由器 J 处在路由器I到路由器K的最优路径上，那么，从路由器J到路由器K的最优路径也在同样的这条路径上。



- 沉落树（ **sink tree** ）：从所有的源到一个给定的目的的最优路径形成的一棵树，树根是目的。

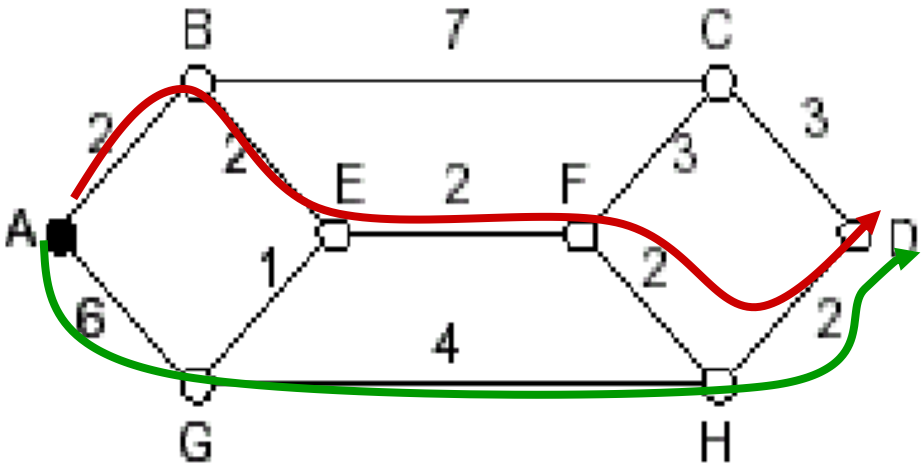
沉落树/汇集树（Sink tree）

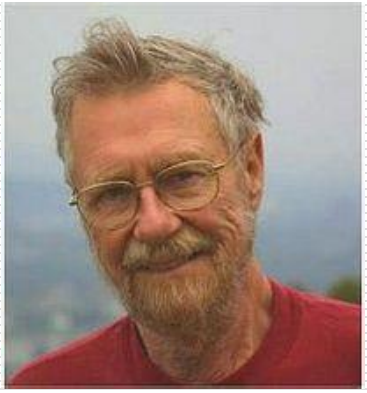
- 汇集树不必是唯一的。
- 所有路由算法的目的就是：为所有的路由器发现和使用汇集树。



最短路径路由选择P282

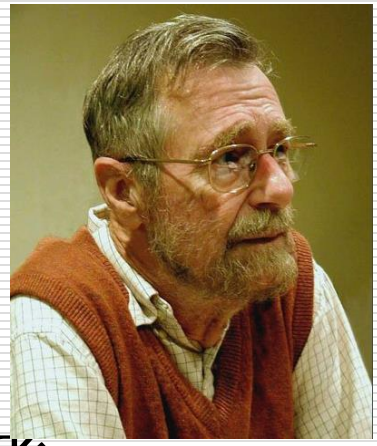
- **Dijkstra 算法（1959）：使用权重（P282）计算通信线路中的最短（优，代价最小）路径。**
- **注意：**
  - **具有最小边数的路径不一定是最短路径**
  - **最短的路径一定是最“快”的路径**



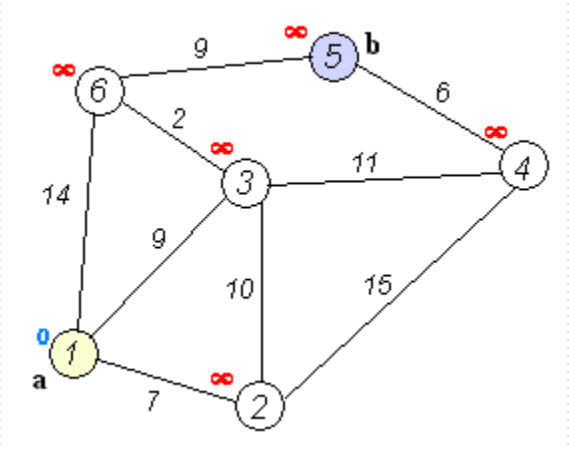


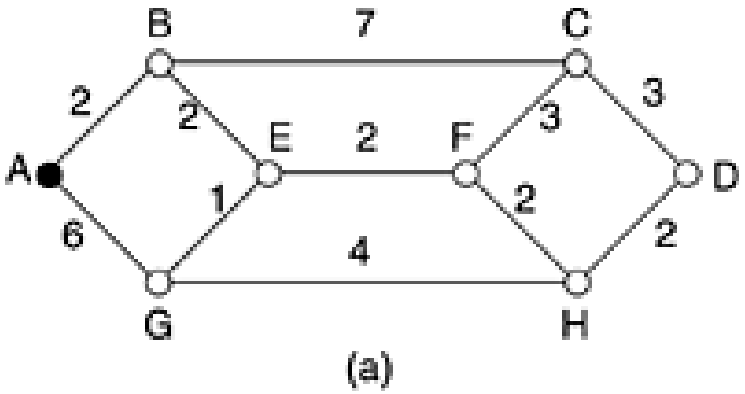
艾兹格·W·迪科斯彻(Edsger Wybe Dijkstra

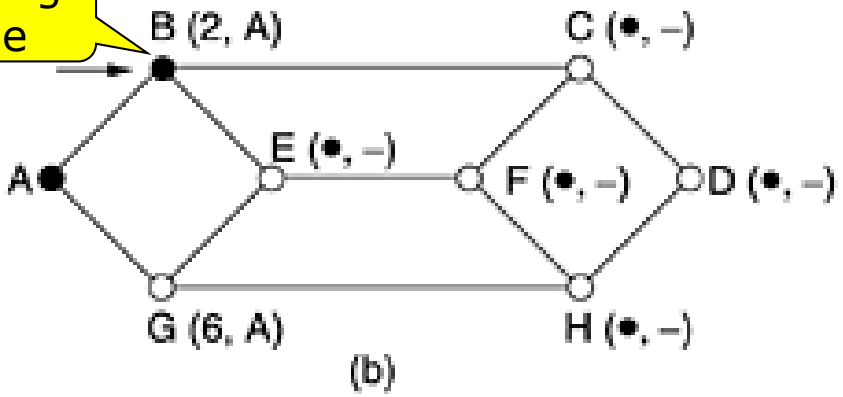
- 1930-2002，数学家，计算机科学家
- 1972年，获图灵奖
- 成就
  - 提出“goto有害论”；
  - 最短路径算法(SPF)和银行家算法的创造者；
  - 第一个Algol 60编译器的设计者和实现者；
  - THE操作系统的设计者和开发者；
  - …….

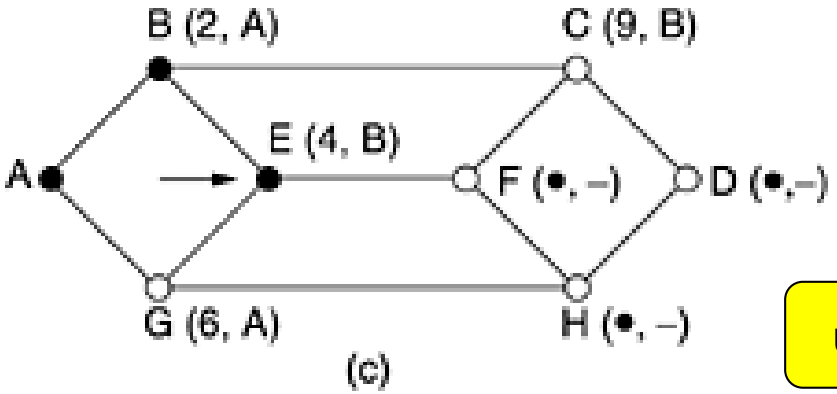


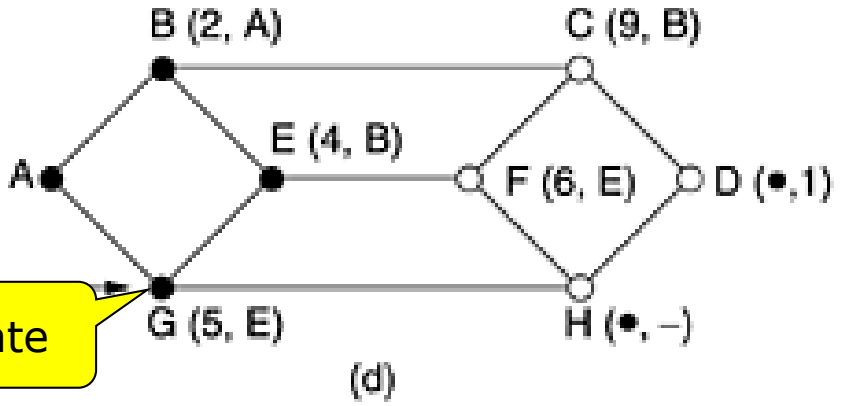
动画演示

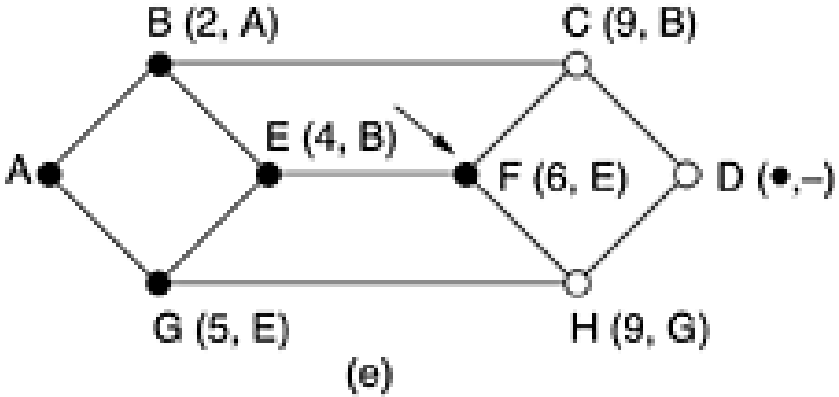


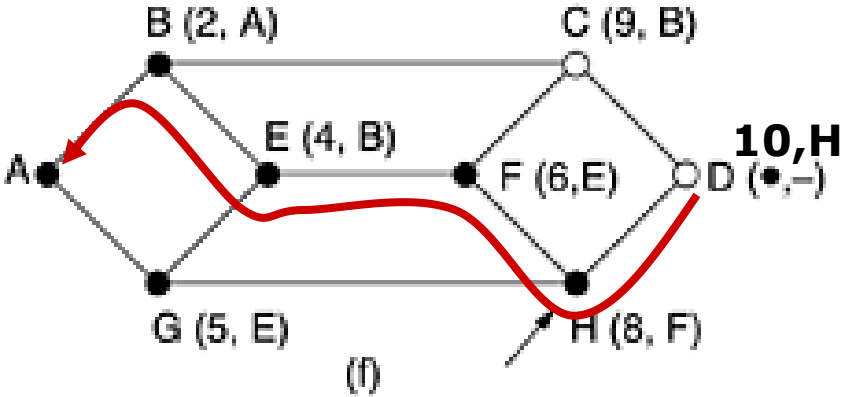

B
7
C
2
2
E
2
F
3
3
A
D
1
2
6
4
2
G
H
(a)


Working node
B (2, A)
C (•, –)
E (•, –)
A
F (•, –)
D (•, –)
G (6, A)
H (•, –)
(b)


B (2, A)
C (9, B)
E (4, B)
A
F (•, –)
D (•,–)
G (6, A)
H (•, –)
(c)


B (2, A)
C (9, B)
E (4, B)
A
F (6, E)
D (•,1)
update
G (5, E)
H (•, –)
(d)


B (2, A)
C (9, B)
E (4, B)
A
F (6, E)
D (•,–)
G (5, E)
H (9, G)
(e)


B (2, A)
C (9, B)
E (4, B)
10,H
A
F (6,E)
D (•,–)
G (5, E)
H (8, F)
(f)

Dijkstra 算法步骤（1/2）

- **1. 初始化**

  假设节点 i 是源节点， N = {i}，对所有不在集合N中的节点：

$$D(v) = \begin{cases} l(i,v) & \text{若v与i直接连接} \\ \infty & \text{若v与i不直接相连} \end{cases}$$

“∞” can be replaced by “any number bigger more than path ”, such as 10^9 (图5-8)

Dijkstra算法步骤（2/2）

- 2. 找到一个不在集合 N 中的节点w ，但它的D(w)最小，把w加入到集合 N 中；那么，所有不在集合N中的节点，使用 min[D(v)，D(w) + l(w,v)] 去替换 D(v):

$$D(v) \leftarrow \min\left[\, D(v),\ D(w)+l(w,v) \,\right]$$

- 3.重复第二步，直到所有的节点都包含在集合N中。

Dijkstra算法程序（1/2）

```
#define MAX_NODES 10[illegible]   /* maximum number of nodes */
#define INFINITY 100000[illegible]  /* number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];     /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                       /* the path being worked on */
      int predecessor;                 /* previous node */
      int length;                      /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p[illegible]) { /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                               /* k is the initial working node */
  do {                                 /* Is there a better path from k? */
```

源

目的

初始化

第一个
工作节点

Dijkstra算法程序（2/2）

```
do {                                    /* Is there a better path from k? */
    for (i = 0; i < n; i++)             /* this graph has n nodes */
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }

    /* Find the tentatively labeled node with the smallest label. */
    k = 0; min = INFINITY;
    for (i = 0; i < n; i++)
        if (state[i].label == tentative && state[i].length < min) {
            min = state[i].length;
            k = i;
        }
    state[k].label = permanent;
} while (k != s);

/* Copy the path into the output array. */
i = 0;  k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

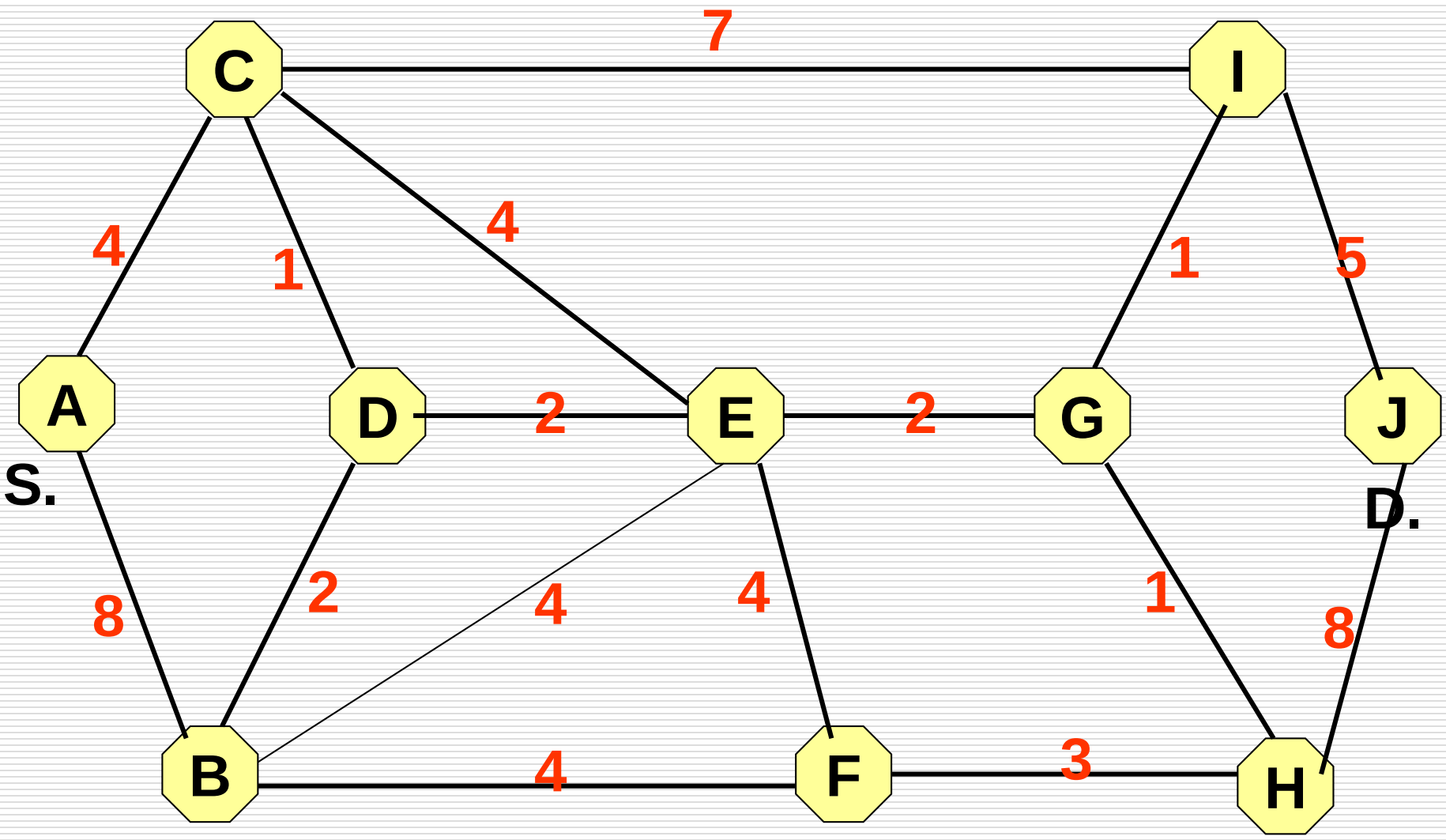
向周边
节点标注

选择下一个
工作节点

根据标注
回溯路径

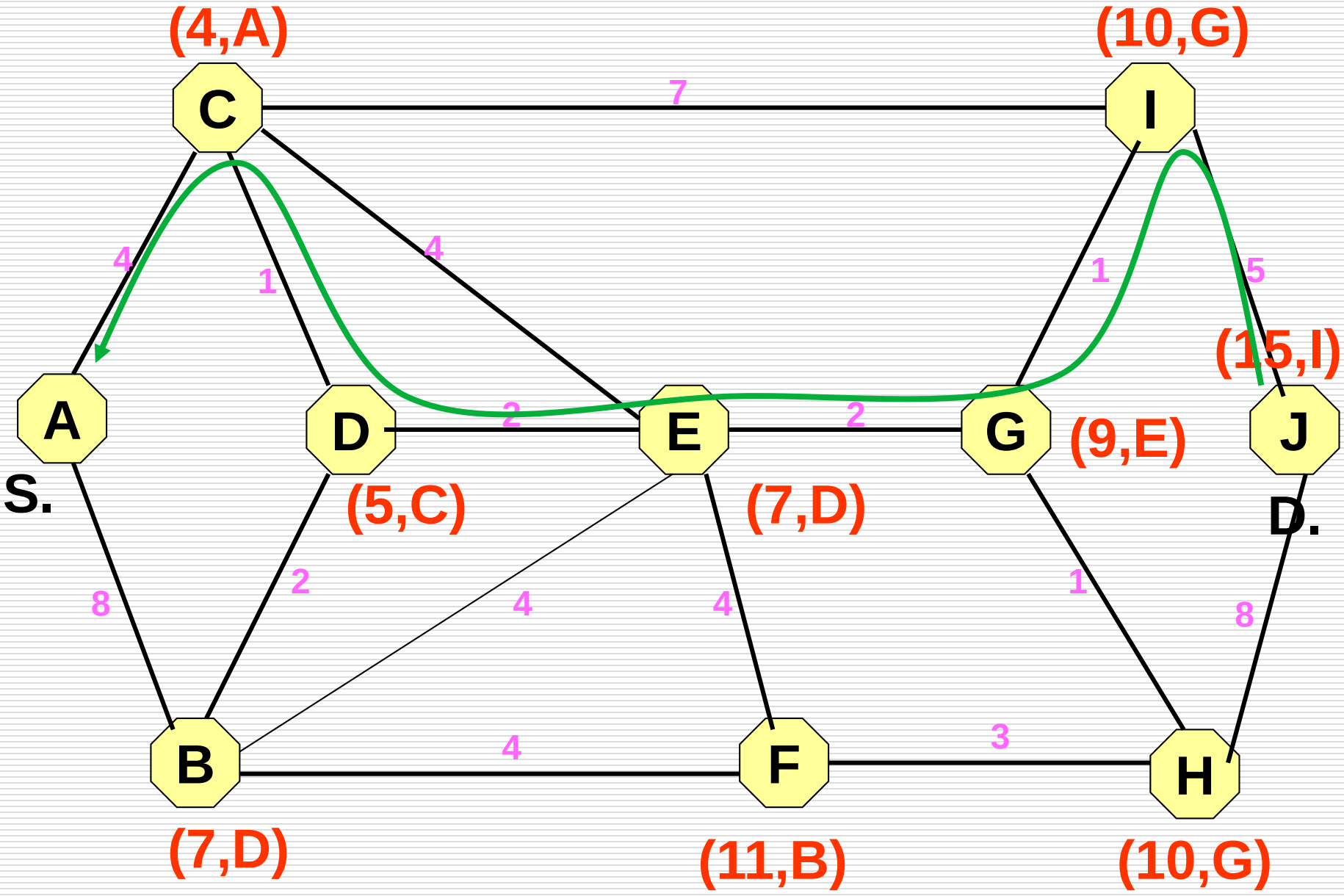
课堂练习1

- 根据下图，试采用Dijkstra 算法计算从A到J的最短路径：
  - 按次序写出工作节点
  - 写出从A到J的最短路径和度量（代价）
  - 遍历完后，每个节点的标注是什么？


C
I
A
D
E
G
J
B
F
H
S.
D.
7
4
1
4
1
5
2
2
8
2
4
4
1
8
4
3

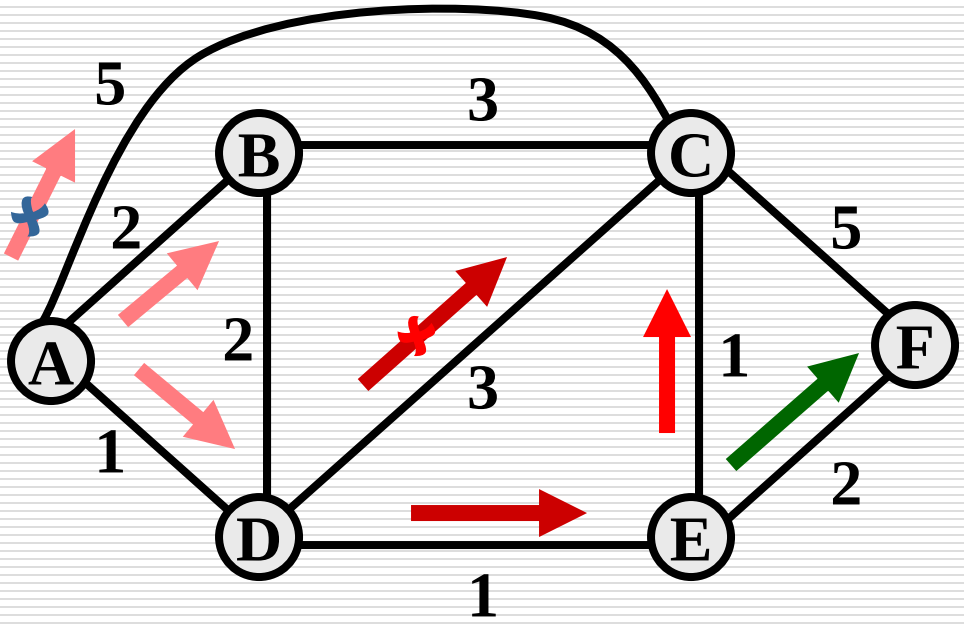
参考答案

- 工作节点：
  - A、C、D、BE、EB、G、IH、HI、F、J
- 最短路径和代价：
  - ACDEGIJ，and cost is 15
- 每个节点的标注如下：


(4,A)
(10,G)
C
I
7
4
4
1
1
5
(15,I)
A
D
E
G
J
2
2
(9,E)
S.
(5,C)
(7,D)
D.
8
2
4
4
1
8
3
B
F
H
4
(7,D)
(11,B)
(10,G)

课堂练习2

- 请根据下列拓扑图，采用Dijkstra算法，求出以 A 为根的沉落树

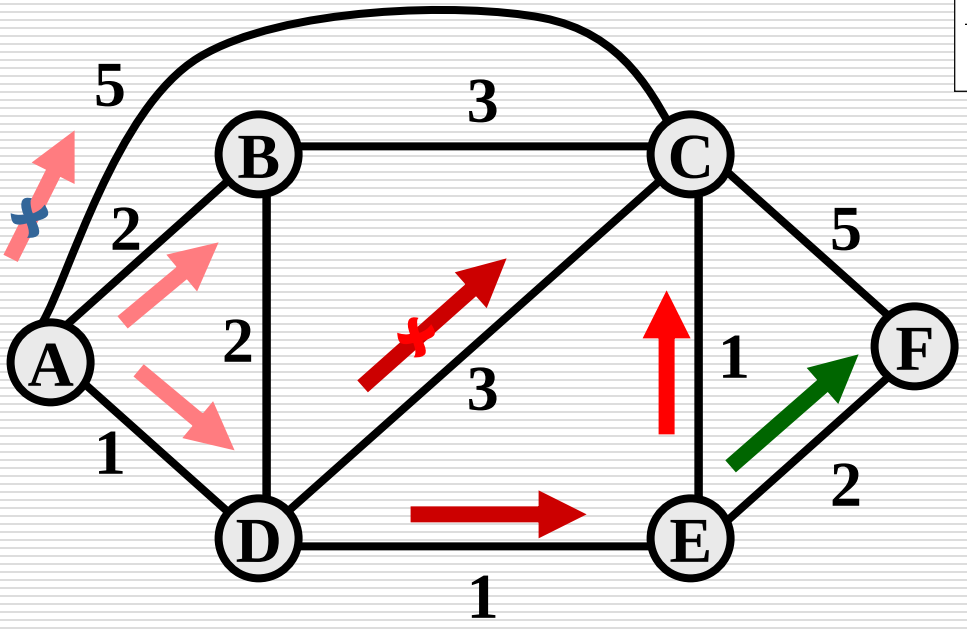


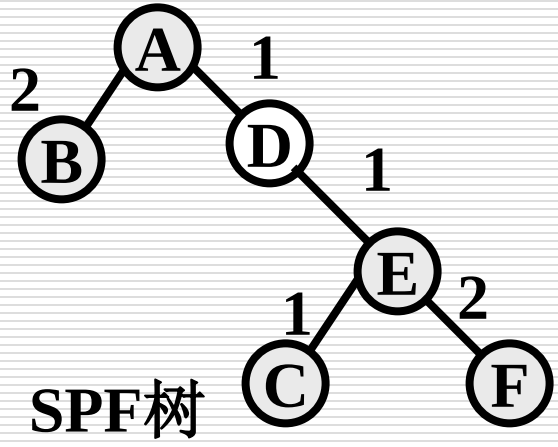
参考答案

源点A到所有结点的最短路径

计算	B	C	D	E	F
0	2,A	5,A	1,A	∞,-	∞,-
1	2,A	4,D		2,D	∞,-
2	2,A	4,D			4,E
3		3,E			4,E
4					4,E

L-S图

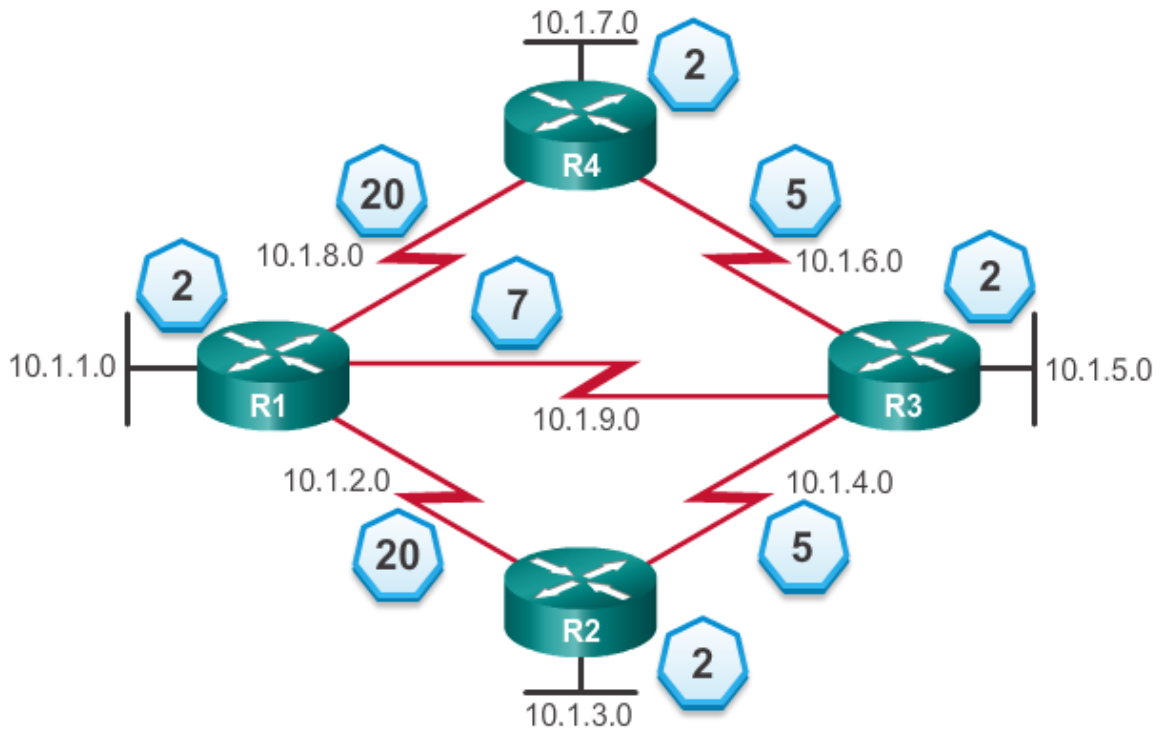




SPF树

课堂练习3

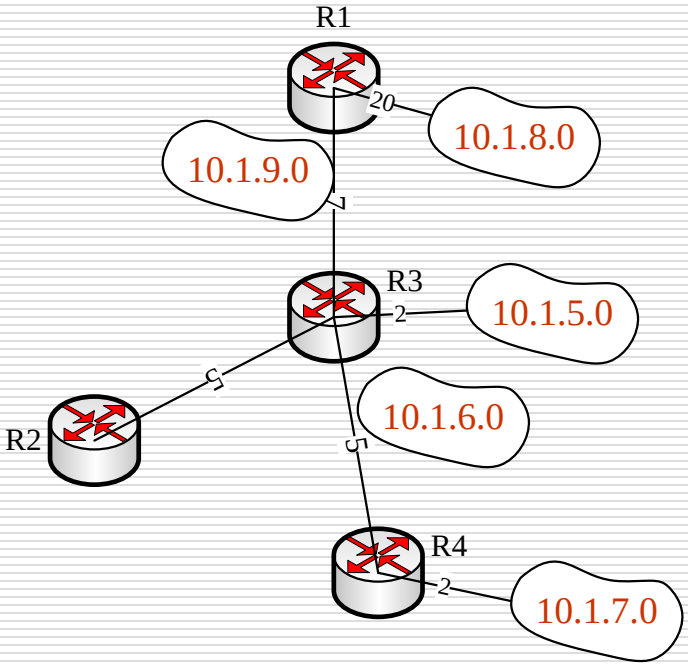
- 绘制R1的SPF树，据此填写路由表项



场景 1

目的网络	开销
10.1.5.0	
10.1.6.0	
10.1.7.0	
10.1.8.0	
10.1.9.0	

参考答案

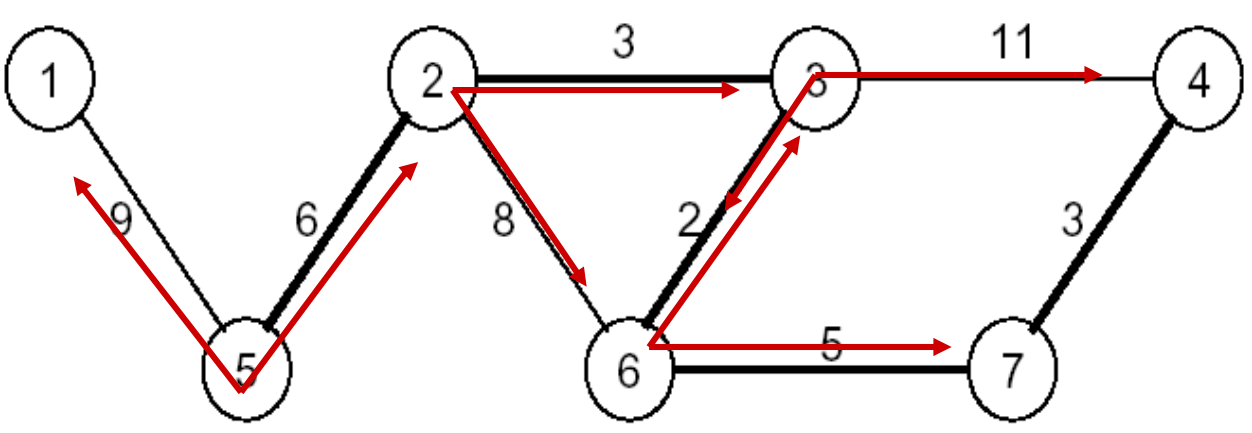


场景 1

目的网络	开销
10.1.5.0	9
10.1.6.0	12
10.1.7.0	14
10.1.8.0	20
10.1.9.0	7

扩散/泛洪法（Flooding，1/2）P284

- 每个到达分组都被从除了到达端口外的所有其它端口转发出去(不计算路径，有路就走)



- 例如，从节点5 到节点 4：packet from 5→1,2；2→3,6；3→6,4；6→3,7；7→4
- 问题：重复分组，例如 3，6

扩散法（Flooding，2/2)

- 解决办法：
  - 在分组头增加一个计数器（counter），每经过一个节点，计算器减 1，当计数器变为零时，报文被丢弃。
  - 每个节点设立一个登记表，当分组第二次到达时，被丢弃。
  - 选择性扩散
- 缺点：重复分组太多，浪费带宽
- 优点：可靠性高、路径最短/优，常用于军事

本节小结

- 网络层的主要功能
  - 选路传输分组 (path selection)
  - 为传输层提供服务
- 路由选择协议
  - Routing-table
- 静态路由选择算法
  - Dijkstra algorithm
  - flooding

Thank you all !