

1. 路由选择算法

1.1 路由的概念

路由：

按照某种指示（传输延迟，所经过的站点数目等）找到一条从源节点到目标节点的较好路径

较好路径：按照某种指标比较小的路径

指标：站数，延迟，费用，队列长度等，或者是一些单纯指标的加权平均

采用什么样的指标，表示网络使用者希望网络在上面方面表现突出，什么指标网络使用者比较重视

以网络为单位进行路由（路由信息通告+路由计算）

网络为单位进行路由，路由信息传输、计算和匹配的代价低

前提条件：一个网络所有节点地址前缀相同，且物理上聚集

路由就是：计算网络到其他网络如何走的问题

概念：

网络-网络的路由=路由器-路由器之间的路由

网络对应的路由器到其他网络对应的路由器的路由

在一个网络中：路由器-主机之间的通信链路层解决

到了这个路由器就是到了这个网络

路由选择算法（**routing algorithm**）：网络层软件的一部分，完成路由功能

图抽象：

$G = (N, E)$

N = 路由器集合 = { u, v, w, x, y, z }

E = 链路集合 = { $(u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z)$ } 边有代价

路由的输入：网络的拓扑、边的代价、源节点

输出：源节点的汇集树

最优化原则：

汇集树：**sink tree**

此节点到所有其他节点的最短路径形成的树

路由选择算法就是为所有的路由器找到并使用汇集树

路由选择算法的原则：

正确性

算法必须是正确的和完整的

正确：使分组一站一站接力，正确发向目标

完整：目标所有的站地址，在路由表中都能找到相应的表项，没有处理不了的目标站地址

简单性

算法在计算机上应简单：最优但复杂的算法，时间上延迟很大，不实用，不应为了获取路由信息增加很多的通信量

健壮性

算法应能适应通信量和网络拓扑的变化

通信量的变化，网络拓扑的变化算法能很快适应

不向很拥挤的链路发送数据，不向断了的链路发送数据

稳定性

产生的路由不应该摇摆

公平性

对每个站点都公平

最优性

对一个指标的最优，时间上，费用上，等指标或综合指标；实际上，获取最优的结果代价较高，可以是

次优的

算法分类：

全局

所有的路由器拥有完整的拓扑和边的代价的信息

link state 算法

分布式

路由器只知道与它有物理连接关系的邻居路由器，和到相应邻居路由器的代价值

迭代地与邻居交换路由信息、计算路由信息

distance vector 算法

静态

路由随时间变化缓慢

非自适应算法（**non-adaptive algorithm**）

不能适应网络拓扑和通信量的变化，路由表是事先计算好的

动态

路由变化的很快

周期性更新

根据链路代价的变化而变化

自适应路由选择（**adaptive algorithm**）能适应网络拓扑的通信量的变化

1.2 link state 路由状态算法（迪杰斯特拉）

配置LS路由选择算法的路由工作过程：

各点通过各种渠道获得整个网络拓扑，网络中所有链路代价等信息（这部分与算法没关系，属于协议和实现）

使用LS路由算法，计算本站点到其他站点的最优路径（汇集树）

按照次路由表转发分组（**datagram**方式）

严格意义上说不是路由的一个步骤

分发到输入端口的网络层

获得网络拓扑和链路代价信息---->使用LS算法得到路由表---->使用路由表

LS路由的基本工作过程：

1.发现相邻节点，获知对方网络地址

一个路由器上电之后，向所有线路发送HELLO分组

其他路由器收到HELLO分组，会送应答，在应答分组中，告知自己的名字（全局唯一）

在LAN中，通过广播HELLO分组，获得其他路由器的信息，可以认为引入一个人工节点

2.测量到相邻节点的代价（延迟，开销）

实测法，发送要分组要求对方立即响应

会送一个ECHO分组

通过测量时间可以估算出延迟情况

3.组装一个LS分组，描述它到相邻节点的代价情况

LS分组：相邻节点以及代价

发送者名称

序号，年龄

列表：给出它相邻节点，和它到相邻节点的延迟

4.将分组通过扩散的方法发到所有其他路由器

泛洪

顺序号：用于控制无穷的扩散，每个路由器都记录（源路由器，顺序号），发现重复的或老的就不扩散

具体问题1：循环使用问题

具体问题2：路由器崩溃之后序号从0开始

具体问题3：序号出现错误

解决问题的办法：年龄字段（age）

生成一个分组时，年龄字段不为0

每隔一个时间段，AGE-1

AGE=0则抛弃该分组

关于扩散分组的数据结构

Source 从那个节点收到LS分组

Seq, Age 序号，年龄

Send flags 发送标记，必须向指定的哪些相邻站点转发LS分组

ACK flags 本站点必须向哪些相邻站点发送应答

DATA 来自source站点的LS分组

以上4步让每个路由器获得拓扑和边代价

5.通过Dijkstra算法找出最短路径（这才是路由算法）

1.每个节点独立算出来到其他节点（路由器=网络）的最短路径

2.迭代算法：第k步能够知道本节点到k个其他节点的最短路径

路由器通过各站点LS分组和整个网络的拓扑
通过Dijkstra算法计算出到其他各路器的最短路径（汇集树）
将计算结果安装到路由表中

LS的应用情况：

OSPF协议是一种LS协议，被用于Internet上

IS-IS（intermediate system-intermediate system）：被用于Internet主干中，Netware

A		Link		State		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

1.3 distance vector 距离矢量算法

动态路由算法之一

DV算法历史以及应用情况

1957 Bellman, 1962 Ford Fulkerson

用于ARPANET, Internet (RIP) DECnet, Novell, AppTalk

距离矢量路由选择的基本思想

各路由器维护一张路由表，结构如图（其他代价）

各路由器与相邻路由器交换路由表

根据获得的路由信息更新路由表

代价及相邻节点间代价的获得

跳数（hops），延迟（delay），队列长度

相邻节点间代价的获得：通过实测

路由信息的更新

根据实测，得到本节点A到相邻站点的代价（如：延迟）

根据各相邻站点声称他们到目标站点B的代价

计算出本站点A经过各相邻点到目标站点B的代价

找到一个最小的代价，和相应的下一个节点Z，到达节点B经过次节点Z，并且代价为A-Z-B

其他所有的目标节点一个计算法

定期测量他到相邻节点的代价----->更新路由表

定期与相邻节点交换路由表DV----->约定次序的往各个目标节点的代价向量：实际为（目标、代价）列表

核心思路：（递归）

每个节点都将自己的距离矢量传输给邻居，定时或DV有变化时，让对方去算

当x从另据收到DV时，自己运算，更新他自己的距离矢量

$$D_x(y) = \min\{c(x, v) + D_v(y)\}$$

$D_x(y)$ 估计值最终收敛于实际的最小代价值 $d_x(y)$

分布式。迭代算法

异步式迭代：

每次本地迭都被一下事件触发

本地链路代价变化

从邻居来了DV的更新消息

分布式：

每个节点只是在自己的DV改变后通知邻居

然后邻居们在有必要的时候同时他们的邻居

每个节点：

等待

本地链路代价变化或者从邻居传送新的DV报文

重新计算

重新计算各目标的代价估量值

通告

如果到任何目标的DV发生变化，通告邻居

DV的无穷计算：

坏消息的传播速度非常慢（无穷计算）

第一次交换后，B从C获得消息，C可以到A（其实是C-B-A），当AB断开连接后，B得知C可以去A（其实不存在），B更新DV，并通知C。C收到后也更新，并通知B，陷入死循环。

解决：

TTL

水平分裂

一种对无穷计算问题的解决办法

C知道要经过B才能到达A，所以C向B报告它到A的距离为INF，C告诉D他到A的真是距离
D告诉E他到A的距离；D告诉C，A不可达

第一次交换：B通过测试发现到A的路径为INF，而C也告诉B到A的距离为INF，因此，B到A的距离为INF

第二次交换：C从B和D那里获知，到A的距离为INF，因此将它到A的距离为INF
坏消息以一次交换一个节点的速度传播

1.4 LS和DV算法的比较

消息复杂度： DV

LS: 有n个节点，E条链路，发送报文O（n*E）个

局部的路由信息，全局传播

DV: 只与邻居交换信息

全局的路由信息，局部传播

收敛时间： LS

LS: O（n^2）

可能震荡

DV: 收敛缓慢

可能存在路由环路

count-to-infinity问题

健壮性：路由器发生故障会发生什么 LS

LS:

节点会通告不正确的链路代价

每个节点只计算自己的路由表

错误信息影响较小，局部，路由较健壮

DV:

DV节点可能通告对全网所有节点的不正确路径代价

距离矢量

每一个节点的路由表可能别其他节点使用

错误可以扩散到全网

