Project 2

Implementing a Distance Vector Routing Protocol

一， 任务要求

分别使用 C/C++、Java 和 Python 实现一个基于 DV 算法的路由选择协议，根据网络状态定期更新路由表（可以不实现分组转发）。

二， 处理要求

启动：

1. 编写的程序是通用的，可以在不同机器、不同操作系统上运行。每次运行视为启动一个节点（即一个路由器）。在一台机器上，可以启动一个或多个节点（即程序可以被多次运行）。在每台机器上启动运行一个节点，可以更加真实地模拟路由协议的工作。每个节点有一个 id。id 为 0、1、2、3、……，也可以为 a、b、c、d、……。启动节点时指定卡启动参数。假设程序名为 dvsim，启动命令如下：

   dvsim 节点 id 节点所使用的 UDP 端口号 节点初始化文件名

   例如：dvsim a 51000 a.txt

   表示：启动一个节点，节点的 id 为 a，节点使用 UDP 端口号 51000 与邻居结点交换路由信息，读取的初始化文件为 a.txt。

   注意：UDP 端口号应与其他节点文本文件中的 UDP 端口号一致。

初始化：

2. 初始启动节点时，每个节点应读取（仅一次）对应该节点的一个初始化文件，用于初始化节点的路由表。该初始化文件为简单的文本文件。节点 a 所对应的文件名字为 a.txt。该文件描述了初始时该节点的所有活动邻居节点及距离或代价。文件的结构如下：

   b 2.0 52001
   d 2.0 52003
   f 2.0 52005

   每个邻居节点占一行。每一行包括 3 个字段，用单空格隔开。每一行表示可以直达节点 y，其距离或代价是 cost，其 UDP 端口号为 portnum。例如第 1 行，b 表示邻居节点的 id，2.0 表示到达该节点的距离或代价，52000 为邻居节点 b 交换路由信息时所使用的 UDP 端口号。

   若在不同的机器上启动节点，且在每台机器上仅启动 1 个节点，各节点可以使用相同的 UDP 端口号。

   文件具体内容需要根据网络拓扑结构自行定义。

3. 初始启动节点时，每个节点还应读取系统的配置文件，获取例如定期时间间隔、不可达距离、最大等待时间等配置参数。

运行：

4. 节点可以一直运行，计算路由并输出计算结果。
5. 节点可以被人为退出运行，例如通过组合键 Ctrl+C 或命令 K。

6. 节点可以被人为暂停运行，例如通过组合键 Ctrl+P 或命令 P，模拟链路或节点故障。可以恢复继续运行，例如通过组合键 Ctrl+S 或命令 S，模拟链路或节点故障修复。

路由表：

7. 节点需要维护路由表，保存从本节点到其他节点的最佳路由。自行定义的路由表必须包括以下成员，其他自行决定：

成员名称	类型	说明
DestNode	String	目的节点 id
Distance	float	到达目的节点的距离或代价。可以不是整数
Neighbor	String	邻居结点 id。若直达，填写邻居结点 id。

交换的路由信息：

8. 邻居节点之间采用 UDP Socket 交换路由信息。
9. 邻居节点之间需要交换路由信息。自行定义的、用于交换路由信息的报文必须包括以下成员，其他自行决定：

成员名称	类型	说明
SrcNode	String	发送节点 id
DestNode	String	目的结点 id
Distance	float	到达目的节点的距离或代价。可以不是整数

10. 邻居节点之间定期交换路由信息。时间以秒为单位，例如每 10 秒。时间在配置文件中设置。若在最大等待时间内（一般为 3 倍的定期时间）收不到邻居节点发送的路由信息，则认为邻居节点或到达邻居节点的链路故障。

输出：

11. 每个节点需在每次将路由信息发送给其邻居时，在屏幕上输出以下结果，并将该结果保存到日志文件中，供后续分析使用。输出结果的格式如下：

Sent. Source Node= node id; Sequence Number = seq
DestNode = node 1’s id; Distance=xx; Neighbor=node’s id （每行表示一条路由）
DestNode = node 2’s id; Distance=xx; Neighbor= node’s id （每行表示一条路由）
DestNode = node 3’s id; Distance=xx; Neighbor= node’s id （每行表示一条路由）
……
其中：

(1) Source Node 为发送节点 id。
(2) Sequence Number 为发送序号。第 1 次发送的序号为 1，第 2 次的序号为 2，依此类推。
(3) DestNode 为目的结点 id。
(4) Distance 为到达目的节点的距离或代价。
(5) Neighbor 为邻居结点 id。（注：需要显示，但交换的路由信息中不包括此字段）

其中字段(3)(4)(5)是重复的，每条路由都包括(3)(4)(5)字段。
你也可以输出或记录更详细的信息，例如时间等。
你也可以输出或记录节点收到的路由信息，例如：
Received. Source Node= node id; Sequence Number = seq
DestNode = node 1’s id; Distance=xx; Neighbor=node’s id （每行表示一条路由）

DestNode = node 2's id; Distance=xx; Neighbor= node's id （每行表示一条路由）
DestNode = node 3's id; Distance=xx; Neighbor= node's id （每行表示一条路由）
……

模拟状态变化：

12. 人为暂停一个节点的运行，例如通过组合键 Ctrl+P 或命令 P，模拟链路或节点故障。恢复被暂停节点继续运行，例如通过组合键 Ctrl+S 或命令 S，模拟链路或节点故障修复。
13. 终止某个/某些节点的运行，例如通过组合键 Ctrl+C 或命令 K。修改这个/这些节点所对应的初始化文件，例如：删除或增加邻居节点，增大或减少到达邻居解决的距离或代价，模拟好消息和坏消息。

三， 系统配置文件与通信记录与结果分析

1. 系统配置文件关键要点

Frequency：定期时间间隔，单位为毫秒。例如：Frequency=1000，表示超时时间为 1 秒。

Unreachable：不可达距离或代价。例如：Unreachable=20.0，表示距离或代价为 20.0 为不可达。

MaxValidTime：最大等待时间，单位为毫秒。例如：MaxValidTime =30000，表示若在 30 秒内为收到邻居节点的路由信息，则邻居节点或到达邻居节点的链路出现故障，邻居节点不活动，为不可达。

2. 通信记录

每个节点在日志文件中记录发送和收到的路由信息，便于事后查看和分析。包括：

Sent. Source Node= node id; Sequence Number = seq
DestNode = node 1's id; Distance=xx; Neighbor=node's id （每行表示一条路由）
DestNode = node 2's id; Distance=xx; Neighbor= node's id （每行表示一条路由）
DestNode = node 3's id; Distance=xx; Neighbor= node's id （每行表示一条路由）
……
Received. Source Node= node id; Sequence Number = seq
DestNode = node 1's id; Distance=xx; Neighbor=node's id （每行表示一条路由）
DestNode = node 2's id; Distance=xx; Neighbor= node's id （每行表示一条路由）
DestNode = node 3's id; Distance=xx; Neighbor= node's id （每行表示一条路由）
……

3. 分析

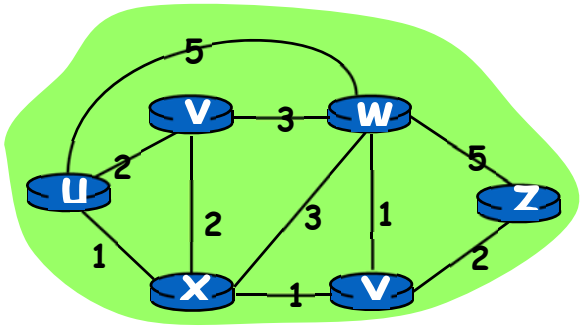
通过设置/配置不同拓扑结构、以及拓扑结构、距离或代价等的变化，分析 DV 的特点：好消息传得快，坏消息传的慢。分析好消息和坏消息情况下的迭代/路由更新次数、是否产生路由回路、路由回路存在时间等。

基于下列 2 个案例，通过分析，利用数据、图表等说明 DV 算法的特点。

在完成下列 2 个案例的分析基础上，你也可以测试其他自行定义的案例。

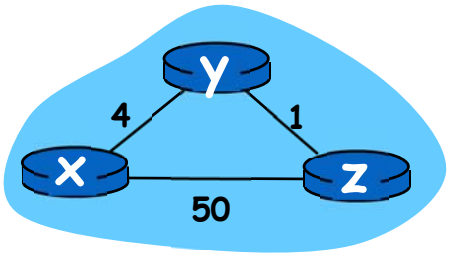
四， 测试案例

案例 1

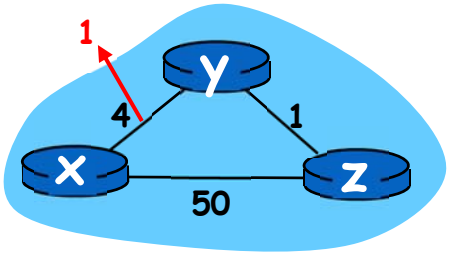


案例 2：

初始：



好消息：



坏消息：