1. 在传输层滑动窗口协议中,A准备发送数据(总共数据大小为20K字节),并向B请求8K字节的缓冲区. B得到了6 K字节的缓冲区,并应答A的发送请求.假设A和B在传送数据时，以1K字节为传送和处理单位(帧)，数据帧从0开始编号。
   * 1. A发送了2K字节(2帧)的数据
     2. A又发送了3K字节(3帧)的数据
     3. B收到5K字节数据,3K-4K间的数据校验出错。系统收回4K分配给B的缓冲区,B对A做否认应答(NACK 3 buf.2K)，A收到B应答
     4. A发送完窗口中能够发送的数据
     5. B正确收到A发送的所有数据,并又从系统多申请了1K缓冲区, B对A做确认应答，A收到B应答

请分别画图说明，发送方(A)窗口在上述每一步骤后的变化情况。

**注意**: 请标明,发送窗口区域、发送并被确认的数据、发送未被确认的数据、可以发送的数据和不能发送的数据。（18分）

1. 对于如下互连网络，假设一个包到达某个网络中就意味着到达了该网络中的所有节点（包括所连接的路由器），也就是说在计算路径距离时，只考虑发送代价不考虑接收代价。设路由表中的每一项具有如下的格式： NetID:Distance:Nexthop。其中NetID表示数据包应该到达的目的网络；Distance表示到达目的网络的距离；Nexthop表示要到达目的网络数据包应该转发到那个路由器上，当不需要转发时Nexthop用“-”表示。网络初启时，各路由器只知道到相连网络的路由信息，例如网络初启时，路由器A的路由表只有如下路由信息：（1：2：-）和（3：2：-）。采用距离向量路由算法的情况下，问：（12分）

（1）仅B和C向邻居发送了自己的路由信息，写出各路由器中关于Net6的路由信息。（4分）

（2）当算法稳定后，写出各路由器中关于Net6的路由信息。（4分）。

（3）步骤2后，路由器C与Net4的线路中断，路由器C关于Net6的路由信息变为（4：∞：-）。此时，仅A，B和C向邻居发送了自己的路由信息，写出各路由器A和B中关于Net6的路由信息。（4分）。



Unfortunately, the question of how long convergence will take is not amenable to quite so simple an answer. Before going any further, it will be useful to look at an example (taken from [2]). Note, by the way, that what we are about to show will not happen with a correct implementation of RIP. We are trying to show why certain features are needed. Note that the letters correspond to gateways, and the lines to networks.

A-----B

\ / \

\ / |

C / all networks have cost 1, except

| / for the direct link from C to D, which

|/ has cost 10

D

|<=== target network

Hedrick [Page 12]

RFC 1058 Routing Information Protocol June 1988

Each gateway will have a table showing a route to each network. However, for purposes of this illustration, we show only the routes from each gateway to the network marked at the bottom of the diagram.

D: directly connected, metric 1

B: route via D, metric 2

C: route via B, metric 3

A: route via B, metric 3

Now suppose that the link from B to D fails. The routes should now adjust to use the link from C to D. Unfortunately, it will take a while for this to this to happen. The routing changes start when B notices that the route to D is no longer usable. For simplicity, the chart below assumes that all gateways send updates at the same time. The chart shows the metric for the target network, as it appears in the routing table at each gateway.

time ------>

D: dir, 1 dir, 1 dir, 1 dir, 1 ... dir, 1 dir, 1

B: unreach C, 4 C, 5 C, 6 C, 11 C, 12

C: B, 3 A, 4 A, 5 A, 6 A, 11 D, 11

A: B, 3 C, 4 C, 5 C, 6 C, 11 C, 12

dir = directly connected

unreach = unreachable

Here's the problem: B is able to get rid of its failed route using a timeout mechanism. But vestiges of that route persist in the system for a long time. Initially, A and C still think they can get to D via B. So, they keep sending updates listing metrics of 3. In the next iteration, B will then claim that it can get to D via either A or C. Of course, it can't. The routes being claimed by A and C are now gone, but they have no way of knowing that yet. And even when they discover that their routes via B have gone away, they each think there is a route available via the other. Eventually the system converges, as all the mathematics claims it must. But it can take some time to do so. The worst case is when a network becomes completely inaccessible from some part of the system. In that case, the metrics may increase slowly in a pattern like the one above until they finally reach infinity. For this reason, the problem is called "counting to infinity".

1. **VC2:H1-A-B-D-H4 ② VC3:H2-B-D-E-H5**

**③ VC1:H1-A-B-E-H5 ④  VC4:H3-C-B-E-H5**

**⑤ VC5:H1-A-B-C-E-H5**

