计时器。若计时器在回复到达前计时就停止了，则信息的发送者可以认定信息已经丢失，并重新发送（如果需要，则一直重复）。通过这种方式，可以避免死锁。

当然，如果原始信息没有丢失，而仅仅是回复延时，接收者可能收到两次或者更多次信息，甚至导致意想不到的结果。想象电子银行系统中包含付款说明的信息。很明显，不应该仅仅因为网速缓慢或者超时设定太短，就重复（并执行）多次。应该将通信规则——通常称为协议（protocol）——设计为让所有事情都正确，这是一个复杂的课题，超出了本书的范围。对网络协议感兴趣的读者可以参考作者的另外一本书——《Computer Networks》(Tanenbaum，2003)。

并非所有在通信系统或者网络发生的死锁都是通信死锁。资源死锁也会发生，如图6-15中的网络。这张图是因特网的简化图（极其简化）。因特网由两类计算机组成：主机和路由器。主机（host）是一台用户计算机，可以是某人家里的PC机、公司的个人计算机，也可能是一个共享服务器。主机由人来操作。路由器（router）是专用的通信计算机，将数据包从源发送至目的地。每台主机都连接一个或更多的路由器，可以用一条DSL线、有线电视连接、局域网、拨号线路、无线网络、光纤等来连接。

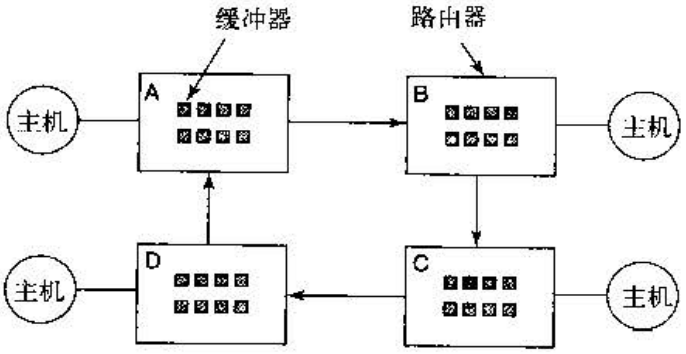

图6-15 一个网络中的资源死锁

当一个数据包从一个主机进入路由器时，它被放入一个缓冲器中，然后传输到另外一个路由器，再到另一个，直至目的地。这些缓冲器都是资源并且数目有限。在图6-15中，每个路由器都有8个缓冲器（实际应用中有数以百万计，但是并不能改变潜在死锁的本质，只是改变了它的频率）。假设路由器A的所有数据包需要发送到B，B的所有数据包需要发送到C，C的所有数据包需要发送到D，然后D的所有数据包需要发送到A。那么没有数据包可以移动，因为在另一端没有缓冲器。这就是一个典型的资源死锁，尽管它发生在通信系统中。

6.7.3 活锁

在某种情形下，轮询（忙等待）可用于进入临界区或存取资源。采用这一策略的主要原因是，相比所做的工作而言，互斥的时间很短而挂起等待的时间开销很大。考虑一个原语，通过该原语，调用进程测试一个互斥信号量，然后或者得到该信号量或者返回失败信息。如图2-26中的例子所示。

现在假设有一对进程使用两种资源，如图6-16所示。每个进程需要两种资源，它们利用轮询原语enter_region去尝试取得必要的锁，如果尝试失败，则该进程继续尝试。在图6-16中，如果进程A先运行并得到资源1，然后进程2运行并得到资源2，以后不管哪一个进程运行，都不会有任何进展，但是哪一个进程也没有被阻塞。结果是两个进程总是一再消耗完分配给它们的CPU配额，但是没有进展也没有阻塞。因此，没有出现死锁现象（因为没有进程阻塞），但是从现象上看好像死锁发生了，这就是活锁（livelock）。

```
void process_A(void) {
        enter_region(&resource_1);
        enter_region(&resource_2);
        use_both_resources( );
        leave_region(&resource_2);
        leave_region(&resource_1);
}

void process_B(void) {
        enter_region(&resource_2);
        enter_region(&resource_1);
        use_both_resources( );
        leave_region(&resource_1);
        leave_region(&resource_2);
}
```

图6-16 忙等待可能导致活锁

活锁也经常出人意料地产生。在一些系统中，进程表中容纳的进程数决定了系统允许的最大进程数量，因此进程表属于有限的资源。如果由于进程表满了而导致一次fork运行失败，那么一个合理的方法是：该程序等待一段随机长的时间，然后再次尝试运行fork。

现在假设一个UNIX系统有100个进程槽，10个程序正在运行，每个程序需要创建12个（子）进程。在每个进程创建了9个进程后，10个源进程和90个新的进程就已经占满了进程表。10个源进程此时便进