4.2 PFC Deadlock

> We once believed that our network is deadlock-free because of its Clos network topology and up-down routing [1, 3, 19]. In such a topology, when a source server sends a packet to a destination server, the packets first climb up to one of the common ancestors of the source and the destination, then go down the path to the destination. Hence there is no cyclic buffer dependency. But to our surprise, we did run into PFC deadlock when we ran a stress test in one of our test clusters. As we will see later, this occurred because the unexpected interaction between PFC and Ethernet packet flooding broke the up-down routing. Before diving into the details of this example, let's briefly review how a ToR switch forwards an IP packet to a server. Typically servers connected to the same ToR are in the same IP subnet. This subnet is then advertised to the rest of the network, so the rest of the network can forward packets to the ToR switch. Once the ToR receives an IP packet which belongs to one of its servers, it needs to query two tables. One is the ARP table from which the ToR switch can figure out the MAC address of the destination server. The second is the MAC address table from which the ToR switch can figure out with which physical port the MAC address is associated. The ARP table is for layer-3 IP whereas the MAC address table is for layer-2. The ARP table is maintained by the ARP protocol. The switch watches which packet comes from which port to establish the MAC address table. Both tables use timeout to retire outdated entries. The typical timeout values for the ARP and MAC tables are very different: 4 hours and 5 minutes, respectively. The reason for using such disparate timeout values is that the overhead of refreshing the entries in the two tables is very different. The MAC table is automati cally refreshed by hardware as new packets are received, while the ARP table is updated by ARP packets, which are handled by the switch CPU. Hence the ARP table maintenance is more costly and thus the ARP table has a much longer timeout value. Such disparate timeout values can lead to an "incomplete" ARP entry – i.e. a MAC address is present in the ARP table, but there is no entry in the MAC address table for that MAC address. When a packet destined to such a MAC address arrives, the switch cannot figure out to which port to forward the packet. The standard behavior in this case is for the switch to flood the packet to all its ports. Below let's use a simplified example as shown in Figure 4 to illustrate how the deadlock happens. We assume all the packets in the example are lossless packets protected by PFC. 1. Server S1 is sending packets to S3 and S5 via path {T0, La, T1}. The purple packets are to S3 and the black packets to S5. S3 is dead, so the purple packets received at port T1.p3 are flooded to the rest ports of T1 including p4. The egress queue of T1.p4 will drop the purple packets once they are at the head of the queue since the destination MAC does not match. But before that, these purple packets are queued there. Also T1.p2 is congested due to incast traffic from S1 and other sources. Hence the black packets are queued in T1. As a result, the ingress port of T1.p3 begins to pause the egress port of La.p1. 2. Consequently, as the black and purple packets build up queues in La, the ingress port of La.p0 begins to pause the egress port of T0.p2. For the same reason, T0.p0's ingress port begins to pause S1. 3. Server S4 begins to send blue packets to S2 via path {T1, Lb, T0}. S2, unfortunately, is also dead. Port T0.p3 then floods the blue packets to the rest ports including T0.p2. Since all packets, including the blue packets, at the egress port of T0.p2 cannot be drained, the ingress port of T0.p3 begins to pause Lb.p0. 4. As a result, the ingress port of Lb.p1 begins to pause T1.p4, and T1.p1 begins to pause S4. Note that T1.p3 will continue to pause La.p1 even if the black packets leave T1 to S5 after the congestion at T1.p2 is gone. This is because the purple packets cannot be drained as T1.p4 is paused by Lb. A PFC pause frame loop among the four switches is then formed. A deadlock therefore occurs. Once the deadlock occurs, it does not go away even if we restart all the servers. This deadlock is a concrete example of the well-known cyclic buffer dependency (see [12, 18, 22, 36] and references therein). The cause of the cyclic dependency, however, is 'new'. It is caused by the flooding behavior of the switch. In an Ethernet switch, once the destination MAC address of a packet is unknown, the packet is Figure 5: The PFC pause frame storm caused by the malfunctioning NIC of one single server (server 0). flooded to all the ports except the receiving port. This 'legitimate' behavior causes the dependency circle to be formed as we have shown in the above example. We need to stop flooding for lossless packets to prevent deadlock from happening. There are several options for us to choose when an ARP entry becomes incomplete (i.e., the IP address to MAC address mapping is there, but the MAC address to port number mapping is not). (1) We forward the packets to the switch CPU and let the switch CPU figure out what to do. (2) We set up the timeout value of the MAC table to be longer than that of the ARP table, so that an ARP entry cannot be incomplete. (3) We drop the lossless packets if their corresponding ARP entry is incomplete. We have chosen option (3). Option (1) may increase the switch CPU overhead. Option (2) needs to either reduce the ARP table timeout value or increase the MAC address table timeout value. If we reduce the ARP table timeout value, we increase the switch CPU overhead for ARP handling. If we increase the MAC address table timeout value, we need longer time to tell when a server becomes disconnected from the switch. Option (3) is a better way to prevent deadlock as it directly prevents the cyclic buffer dependency. The lesson we have learned from the PFC deadlock is that broadcast and multicast are dangerous for a lossless network. To prevent deadlock from happening, we recommend that broadcast and multicast packets should not be put into lossless classes.

我们曾经相信我们的网络是无死锁的，因为它的 Clos 网络拓扑和上下路由 [1,3,19]。在这种拓扑中，当源服务器将数据包发送到目标服务器时，数据包首先爬升到源服务器和目标服务器的共同祖先之一，然后沿着路径向下到达目标服务器。因此不存在循环缓冲区依赖性。
但令我们惊讶的是，当我们在其中一个测试集群中运行压力测试时，我们确实遇到了 PFC 死锁。正如我们稍后将看到的，发生这种情况是因为 PFC 和以太网数据包洪泛之间的意外交互破坏了上下路由。

在深入了解此示例的详细信息之前，我们先简要回顾一下 ToR 交换机如何将 IP 数据包转发到服务器。
通常，连接到同一 ToR 的服务器位于同一 IP 子网中。然后将该子网通告到网络的其余部分，以便网络的其余部分可以将数据包转发到 ToR 交换机。
一旦 ToR 收到属于其一台服务器的 IP 数据包，它就需要查询两个表: 第一个是 ARP 表，ToR 交换机可以从该表中找出目标服务器的 MAC 地址。第二个是 MAC 地址表，ToR 交换机可以从中找出该 MAC 地址与哪个物理端口关联。 ARP 表用于第 3 层 IP，而 MAC 地址表用于第 2 层。 ARP 表由 ARP 协议维护。交换机监视哪个数据包来自哪个端口，以建立 MAC 地址表。
两个表都使用超时来淘汰过时的条目。 ARP 和 MAC 表的典型超时值非常不同：分别为 4 小时和 5 分钟。
使用如此不同的超时值的原因是刷新两个表中的条目的开销非常不同。
当收到新数据包时，MAC 表会由硬件自动刷新; 而 ARP 表则由 ARP 数据包更新，由交换机 CPU 处理。因此 ARP 表的维护成本更高，因此 ARP 表的超时值更长。
这种不同的超时值可能会导致"不完整"的 ARP 条目，即 ARP 表中存在 MAC 地址，但 MAC 地址表中没有该 MAC 地址的条目。*当发往此类 MAC 地址的*

数据包到达时，交换机无法确定将数据包转发到哪个端口(因为MAC地址表提供的物理端口位置缺失)。=> 在这种情况下，标准行为是交换机将数据包洪泛到其所有端口。

Below let's use a simplified example as shown in Figure 4 to illustrate how the deadlock happens. We assume all the packets in the example are lossless packets protected by PFC

1. 服务器 S1 通过路径 {T0, La, T1} 向 S3 和 S5 发送数据包。紫色数据包发送至 S3，黑色数据包发送至 S5。 S3 已失效，因此在端口 T1.p3 收到的紫色数据包将被洪泛到 T1 的其余端口（包括 p4）【*一旦数据包的目的MAC地址未知，数据包就会被洪泛到除接收端口之外的所有端口*】。
2. 一旦紫色数据包位于队列头部，T1.p4 的*出口队列*就会丢弃它们，因为目标 MAC 不匹配。但在此之前，这些紫色数据包就在那里排队。
3. 此外，由于来自 S1 和其他来源的组播流量，T1.p2 也发生拥塞。因此，黑色数据包在 T1 中排队。结果，*T1.p3的入口端口开始暂停La.p1的出口端口*。
4. 因此，当黑色和紫色数据包在 La 中建立队列时，*La.p0 的入口端口*开始暂停 *T0.p2 的出口端口*。出于同样的原因，*T0.p0 的入口端口开始暂停 S1*【*s1是提供方，收到pfc后要暂停*】。
5. 服务器S4开始通过路径{T1，Lb，T0}向S2发送蓝色数据包。不幸的是，S2也死了。然后，*端口 T0.p3【是接收端口】* 将蓝色数据包泛洪到包括 T0.p2 在内的其余端口。由于T0.p2出端口的所有数据包（包括蓝色数据包）无法被排出【因为：上述4.中*La.p0 的入口端口*开始暂停 *T0.p2 的出口端口*】，因此*T0.p3的入端口开始暂停Lb.p0*。
6. 结果，*Lb.p1的入端口开始暂停T1.p4*，T1.p1开始暂停S4。
   请注意，即使在 T1.p2 的拥塞消失后黑色数据包离开 T1 到 S5，T1.p3 也会继续暂停 La.p1。这是因为 T1.p4 被 Lb 暂停，因此紫色数据包无法被排出。然后在四个开关之间形成PFC暂停帧环路。因此出现死锁。一旦发生死锁，即使我们重新启动所有服务器，死锁也不会消失。
   这种死锁是众所周知的循环缓冲区依赖性的具体示例（参见[12,18,22,36]及其中的参考文献）。然而，循环依赖的原因是"新的"。这是由交换机的洪泛行为引起的。在以太网交换机中，一旦数据包的目的MAC地址未知，数据包就会被洪泛到除接收端口之外的所有端口。正如我们在上面的示例中所示，这种"合法"行为会导致依赖圈的形成。

我们需要停止无损数据包的泛洪，以防止死锁的发生。
当 ARP 条目不完整时（即 IP 地址到 MAC 地址的映射存在，但 MAC 地址到端口号的映射不存在），我们可以选择多种选项。
(1) 我们将数据包转发给交换机CPU，让交换机CPU决定要做什么。
(2) 我们将MAC表的超时值设置得比ARP表的超时值长，这样就不会出现ARP表项不完整的情况。
(3) 如果对应的 ARP 条目不完整，我们将丢弃无损数据包。
我们选择了选项（3）。选项（1）可能会增加交换机CPU开销。选项（2）需要减少 ARP 表超时值或增加 MAC 地址表超时值。如果我们减少 ARP 表超时值，则会增加用于 ARP 处理的交换机 CPU 开销。如果我们增加 MAC 地址表超时值，我们需要更长的时间来判断服务器何时与交换机断开连接。选项（3）是防止死锁的更好方法，因为它直接防止循环缓冲区依赖性。我们从 PFC 死锁中学到的教训是，广播和组播对于无损网络来说是危险的。为了防止死锁的发生，我们建议不要将广播和组播数据包放入无损类。

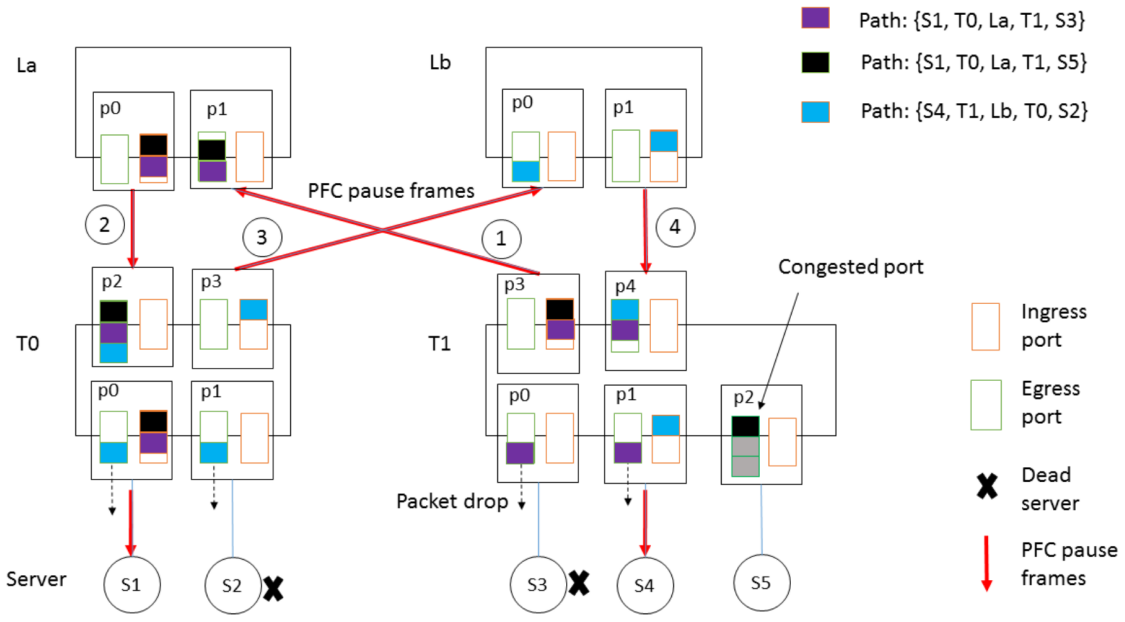

Figure 4: An example to show that the interaction between Ethernet packet flooding and PFC pause frame propagation can cause deadlock.