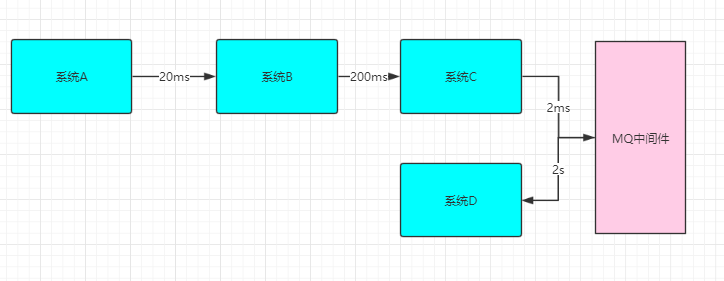
一: **系统可用性降低**

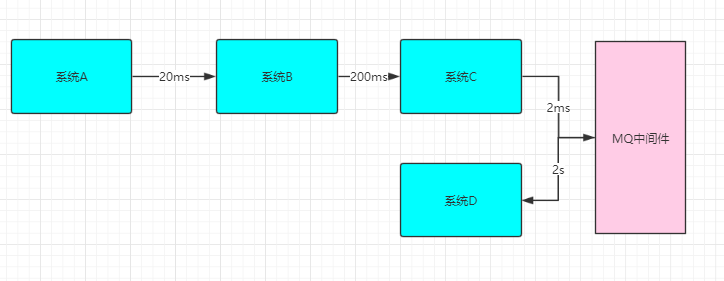


但是你有没有考虑另外一个问题，就是万一你依赖的那个MQ中间件突然挂掉了怎么办？这个还真的不是异想天开，MQ、Redis、MySQL这些组件都有可能会挂掉。

一旦你的MQ挂了，就导致你的系统的核心业务流程中断了。本来你要是不引入MQ中间件，那其实就是一些系统之间的调用，但是现在你引入了MQ，就导致你多了一个依赖。一旦多了一个依赖，就会导致你的可用性降低。

因此，一旦引入了MQ中间件，你就必须去考虑这个MQ是如何部署的，如何保证高可用性。就算保证了高可用，也要保证MQ宕机后的降级处理方案。

二: **系统稳定性降低**



比如说，莫名其妙的，系统C发了一个消息到MQ，结果那个消息因为网络故障等问题，就丢失了。这就导致系统D没有收到那条消息。这可就惨了，这样会导致系统D没完成自己该做的任务，此时可能整个系统会出现业务错乱，数据丢失，严重的bug，用户体验很差等各种问题。

这还只是其中之一，万一说系统C给MQ发送消息，不小心一抽风重复发了一条一模一样的，导致消息重复了，这个时候该怎么办？可能会导致系统D一下子把一条数据插入了两次，导致数据错误，**脏数据的产生**，最后一样会导致各种问题。

或者说如果系统D突然宕机了几个小时，导致无法消费消息，结果大量的消息在MQ中间件里积压了很久，这个时候怎么办？即使系统D恢复了，也需要慢慢的消费数据来进行处理。

三: **分布式一致性问题**

举个例子，比如说系统C现在处理自己本地数据库成功了，然后发送了一个消息给MQ，系统D也确实是消费到了。

但是结果不幸的是，系统D操作自己本地数据库失败了，那这个时候咋办？

系统C成功了，系统D失败了，会导致系统整体数据不一致了啊。

所以此时又需要使用可靠消息最终一致性的分布式事务方案来保障。