IOCP中在WSASend以及WSARecv的时候出现WSA\_IO\_PENDING情况的说明

在IOCP中投递WSASend返回WSA\_IO\_PENDING的时候，表示异步投递已经成功，但是稍后发送才会完成。这其中涉及到了三个缓冲区：

网卡缓冲区，TCP/IP缓冲区，应用程序缓冲区。

情况一：调用WSASend发送正确的时候（即立即返回，且没有错误），TCP/IP将数据从应用程序缓冲区中拷贝到TCP/IP缓冲区中，然后不锁定该应用程序缓冲区，由上层程序自己处理。TCP/IP层缓冲区在网络合适的时候，将其数据拷贝到网卡缓冲区，进行真正的发送。（同步方式）

情况二：调用WSASend发送错误，但是错误码是WSA\_IO\_PENDING的时候，表示此时TCP/IP缓冲区已满，暂时没有剩余的空间将应用程序缓冲区的数据拷贝出来，这时系统将锁定用户的程序缓冲区， WSASend指定的缓冲区将会被锁定到系统的非分页内存中。直到TCP/IP缓冲区有空余的空间来容纳程序缓冲区中的数据，并将给IOCP一个完成消息。（异步方式）

情况三：调用WSASend发送错误，但是错误码不是WSA\_IO\_PENDING，此时应该是发送错误，应该释放该SOCKET及其相关资源。

在IOCP中投递WSARecv的时候，情况相似：

情况一：调用WSARecv正确，TCP/IP将数据从TCP/IP缓冲区拷贝到程序缓冲区，然后由我们的程序自行处理了。同时TCP/IP还会清除TCP/IP缓冲区中已拷贝的数据。

情况二：调用WSARecv错误，但是返回值是WSA\_IO\_PENDING，此时是因为TCP/IP缓冲区中没有数据可取，系统将会锁定我们投递的WSARecv的buffer，直到TCP/IP缓冲区中有新的数据到来。

情况三：调用WSARecv错误，错误值不是WSA\_IO\_PENDING，此时是接收出错，应该释放该SOCKET及其相关资源。

在以上情况中有几个非常要注意的事情：

系统锁定非分页内存的时候，最小的锁定大小是4K(当然，这个取决于您系统的设置，也可以设置小一些，在注册表里面可以改，当然我想这些数值微软应该比我们更知道什么合适了)，所以当我们投递了很多WSARecv或者WSASend的时候，不管我们投递的Buffer有多大（0除外），系统在出现IO\_PENGDING的时候，都会锁定我们4K的内存。这也就是经常有开发者出现WSANOBUF的情况原因了。

我们在解决这个问题的时候，要针对WSASend和WSARecv做处理

1投递WSARecv的时候，可以采用一个巧妙的设计，先投递0大小Buffer的WSARecv，如果返回，表示有数据可以接收，我们开启真正的recv将数据从TCP/IP缓冲区取出来，直到WSA\_IO\_PENGDING.

2对投递的WSARecv以及WSASend进行计数统计，如果超过了我们预定义的值，就不进行WSASend或者WSARecv投递了。

3现在我们应该就可以明白为什么WSASend会返回小于我们投递的buffer中的数据大小了，是因为TCP/IP层缓冲区小于我们要发送的缓冲区，TCP/IP只会拷贝他剩余可被Copy的缓冲区大小的数据走，然后给我们的WSASend的已发送缓冲区设置为移走的大小，下一次投递的时候，如果TCP/IP层还未被发送，将返回WSA\_IO\_PENGDING。

4在很多地方有提到，可以关闭TCP/IP层缓冲区，可以提高一些效率和性能，这个从上面的分析来看，有这个可能，要实际的网络情况去实际分析了。