There is a potential for deadlock when allocating a struct sk_buff for data that needs to be written out to aoe storage. If the data is being written from a dirty page in order to free that page, and if there are no other pages available, then deadlock may occur when a free page is needed for the sk_buff allocation. This situation has not been observed, but it would be nice to eliminate any potential for deadlock under memory pressure.

Because ATA over Ethernet is not fragmented by the kernel's IP code, the destructor member of the struct sk_buff is available to the aoe driver. By using a mempool for allocating all but the first few sk_buffs, and by registering a destructor, we should be able to efficiently allocate sk_buffs without introducing any potential for deadlock.