**Exer. 10-1.**Deadlocks are possible if there is no timeout. In a two-way handshake, the client would send the SYN packet, and the server would respond with the SYN-ACK -- no ACK would have to be sent from the client. If we consider the scenario where A sends a SYN packet to B, and B sends a SYN packet to A. A would then receive the SYN packet, and send a SYN-ACK to B. Likewise, B would receive the SYN packet, and send a SYN-ACK to A. As a result, A would not be able to determine if the SYN-ACK they received is a response to their outgoing SYN, or if it's B's response to A's SYN-ACK. Since there is no timeout, both sides would wait indefinitely, causing a deadlock. This is actually the reason why TCP uses a three-way handshake. By requiring the client to send a final ACK, confirming that they received the SYN-ACK from the server, this ambiguity is avoided, and the order of requests is made clear.

**Exer. 10-2.**The first reason why UDP and TCP protocols invented a new abstract ID (port numbers) instead of using process IDs, which already existed, is because process IDs are assigned dynamically by operating system. This means that process IDs could be different across different operating systems. Furthermore, the same process could have different process IDs each time it runs, making it essentially impossibly to reliably target a specific service. The second reason is because port numbers are standardized and they provide a layer of abstraction. For instance, a well-known port like 80, regardless of the operating system, signifies an HTTP service. Additionally, multiple processes can listen to a given port number at different times. Ultimately, this creates a reliable method to addressing services, such as HTTP or SSH, regardless of which process is handling them.

--Jason McCauley and Deep Shah