scaling strategy of Darkstar. The session is also responsible for ensuring that the order of messages is maintained. A message from a given client will not be delivered if the tasks that resulted from previous message deliveries have not completed. Having the session service order tasks in this way significantly simplifies the Task Service, which can assume that all of the tasks that it has at any time are essentially concurrent. The ordering of messages from a particular client is the only message-ordering guarantee made within the Darkstar framework; external observers might see an ordering of messages from multiple clients that is very different from that seen within the game or virtual world.

The second communication service that is always available in the Darkstar stack is the Channel Service. Channels are a form of one-to-many communication. Conceptually, channels can be joined by any number of clients, and any message that is sent on the channel will be delivered to all of the clients that have been associated with the channel. This might seem to be a perfect place to utilize peer-to-peer technologies, allowing clients to directly communicate with other clients without adding any load to the server. However, these sorts of communications need to be monitored by some code that is trusted to ensure that neither inappropriate messages nor cheating can take place by utilizing different client implementations. Since the client is assumed to be under the control of the user or player, the code that is on that client cannot be trusted, because it is easy to swap out the original client code for some other, "customized" version of the client. So, in fact, all channel messages have to go through the server, after being (possibly) vetted by the server logic.

One of the complexities of both Sessions and Channels is that they must obey the transactional semantics of tasks. Thus the actual transmission of a message on either a Session link or a Channel cannot happen when the call is made to the appropriate `send()` method; it can happen only when the task in which that method occurs commits.

Supplying these communication mechanisms gives us some of the pieces that are needed for the second part of our scaling mechanism. Since all communication must go through the Darkstar Session or Channel abstractions, and since those abstractions do not reveal the actual endpoints of the communication to the client or the server, there is a layer of abstraction between the entities communicating and the actual locations that are the start and end to that communication. This means that we can move the endpoint of the server communication from one machine in the Darkstar system to another without changing the way the client views the communication. From the client's point of view, all communication happens on a particular session or channel. From the point of view of the game or virtual world logic, communication is also through a single session or channel. But the underlying infrastructure can move the session or channel from one machine to another as needed to balance load as that load changes over time.