Actor Owner and Owning Connection

Actor owner, owning connection, and what this tells you about an actor in networked gameplay.

There are many different types of parent-child relationships for objects in Unreal Engine. Two important relationships for network replication in Unreal Engine are an actor's owner and the owner's associated owning connection.

Overview

Multiplayer in Unreal Engine uses a server-authoritative, client-server model. In this model, clients connect to a centralized server. When a client connects to the server, a player controller is created on the server associated with the connected client (a connected client is referred to as a *connection*). When the client begins play on the server, this player controller possesses a pawn that the client controls in the game. The player controller is the *owner* of the pawn. The *owning connection* of an actor is the connection associated with the actor's owning player controller. The owner and owning connection determine which connected client has the authority to make changes and call remote functions.

Every `AActor`-derived object stores a pointer to its owner. Not every `AActor`-derived object has an owner. The actor's owner may be null, in which case the actor has no owner.

Uses of Owning Connection

Connection ownership is important for:

- Actor replication
- Property replication
- RPCs

An actor's owning connection is used during actor replication to determine which connections get updates for an actor, known as actor relevancy. For actors that have `bOnlyRelevantToOwner` set to true, only the connection that owns the actor receives property updates for the actor. By default, all player controllers are only relevant to their owner. This is why each client only receives updates for their own player controllers.

An actor's owning connection is used during property replication involving conditions that use the owner. For more information about how an actor's owner can affect property replication, see Conditional Replication.

An actor's owning connection is also important for RPCs. When the server calls a client RPC function on an actor, unless the RPC is marked as `NetMulticast`, the RPC needs to know which connections to execute the RPC on. The actor's owning connection determines the connections to send and execute the RPC.

Determine Owner and Owning Connection

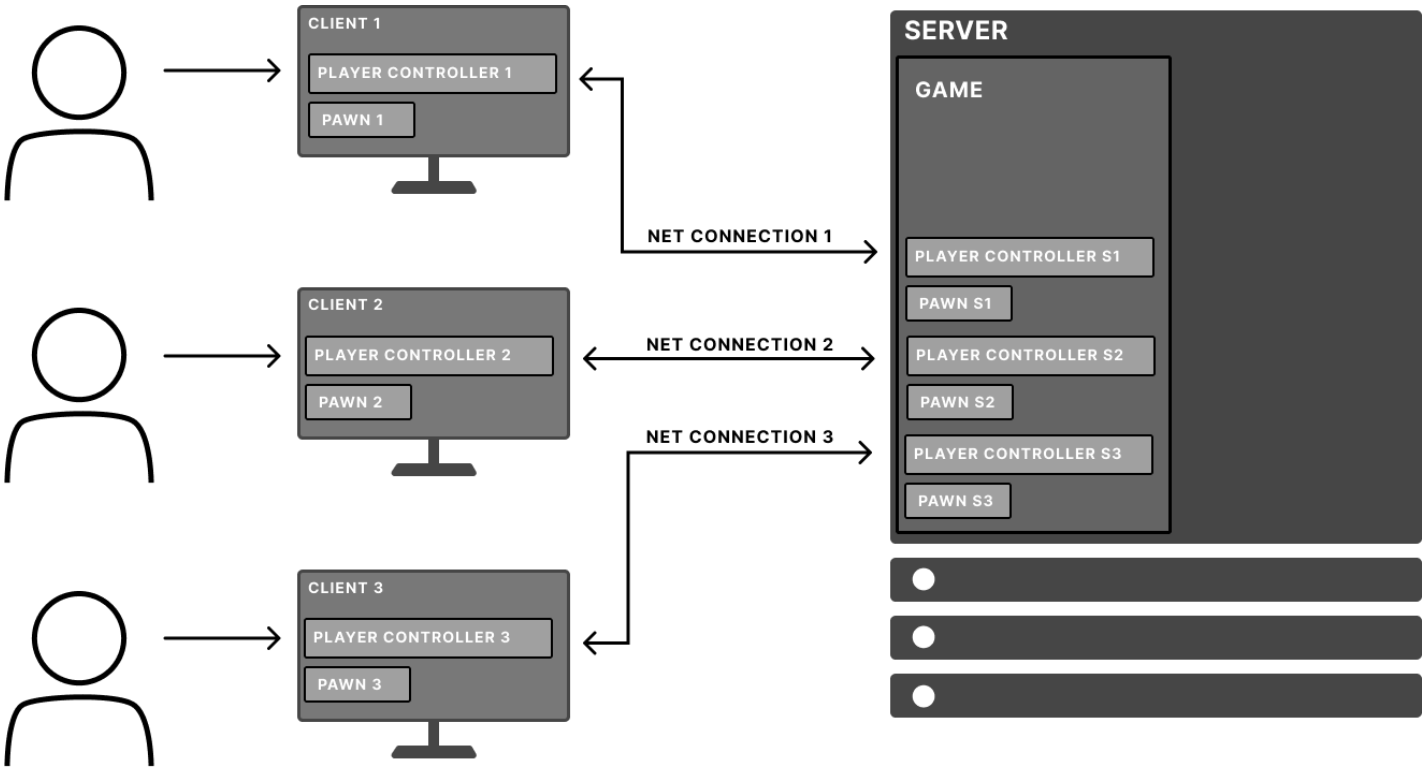

Figure 1. Clients connected to a central server showing pawns, owning player controllers, and owning connections.

Suppose you are playing a multiplayer game as a client connected to the server. Your player controller on your machine is the abstraction of you as a player. In the context of *Figure 1*, if you are playing on Client 1, your input is processed in Player Controller 1 and communicated to Pawn 1. As previously mentioned, when your client machine connects to the server, Net Connection 1 is established, and Player Controller S1 is created on the server associated with your connection. The server pawn actor, Pawn S1, is possessed by Player Controller S1. This means that Player Controller S1 owns Pawn S1. The owning connection of Pawn S1 is Net Connection 1, which is the owning connection for Player Controller S1. Pawn S1 is only owned by this connection during the time it is also owned/possessed by Player Controller S1. As soon as Player Controller S1 no longer possesses Pawn S1, Net Connection 1 is no longer the owning connection for Pawn S1.

The same logic applies to inventory items that may belong to your in-game character pawn. Inventory items are owned by the same connection that might own the pawn.

Actor components operate in a similar fashion to determine ownership but with some additional steps. For actor components, you must first determine the component's owner by walking the actor component's outer chain until the owning actor is found. You can follow the process outlined above to determine the owning connection for the actor component's owning actor.

Owner

To determine the owner of an actor, you query for an actor's outermost owner. If the outermost owner is a player controller, then the original actor's owning connection is the same as the player controller's owning connection.

To obtain an actor's owner, call `AActor::GetOwner`.

To obtain an actor component's owning actor, call `UActorComponent::GetOwner`.

Owning Connection

To obtain an actor's owning connection, call `AActor::GetNetConnection`.