Performance and Bandwidth Tips

Some tips for optimizing the performance and bandwidth usage of Actor replication.

Replicating Actors can take time. **Unreal Engine (UE)** does its best to make this as efficient as possible, but there are a few things you can do to make this job easier.

When gathering actors for replication, the server will check a few things like relevancy, update frequency, dormancy, etc. You can tweak any of these checks to affect performance. When thinking about making this process as efficient as possible, it is best to prioritize in this order:

- Turning off replication (`AActor::SetReplicates(false)`)
  - When an actor is not replicating, it is not on the list in the first place, so this is the biggest win, to make sure actors that do not need to replicate are marked as such.
- Lower `NetUpdateFrequency` value
  - The less an actor updates, the less time it takes to update. It is best to make this number as low as possible. This number represents how often per second this actor will replicate to clients.
- Dormancy
- Relevancy
- `NetClientTicksPerSecond`

Do not mark properties to replicate if they are not absolutely necessary. It is best to try and derive state from existing replicated properties when possible.

Try to take advantage of the quantization functionality that already exists. e.g. `FVector_NetQuantize`. These will greatly reduce the size needed to replicate this state over to clients, and if used properly, should not cause any noticeable artifacts.

`FName` variables are not generally compressed, so when you are using them as parameters to RPCs, keep in mind that they will generally send the string each call. This can be a lot of overhead.