

- Developer
- / Documentation
- / Unreal Engine ▾
- / Unreal Engine 5.4 Documentation
- / Making Interactive Experiences
- / Networking and Multiplayer
- / Testing, Debugging, and Optimization
- / Network Emulation

# Network Emulation

Emulate network packet lag and loss in Unreal Engine.



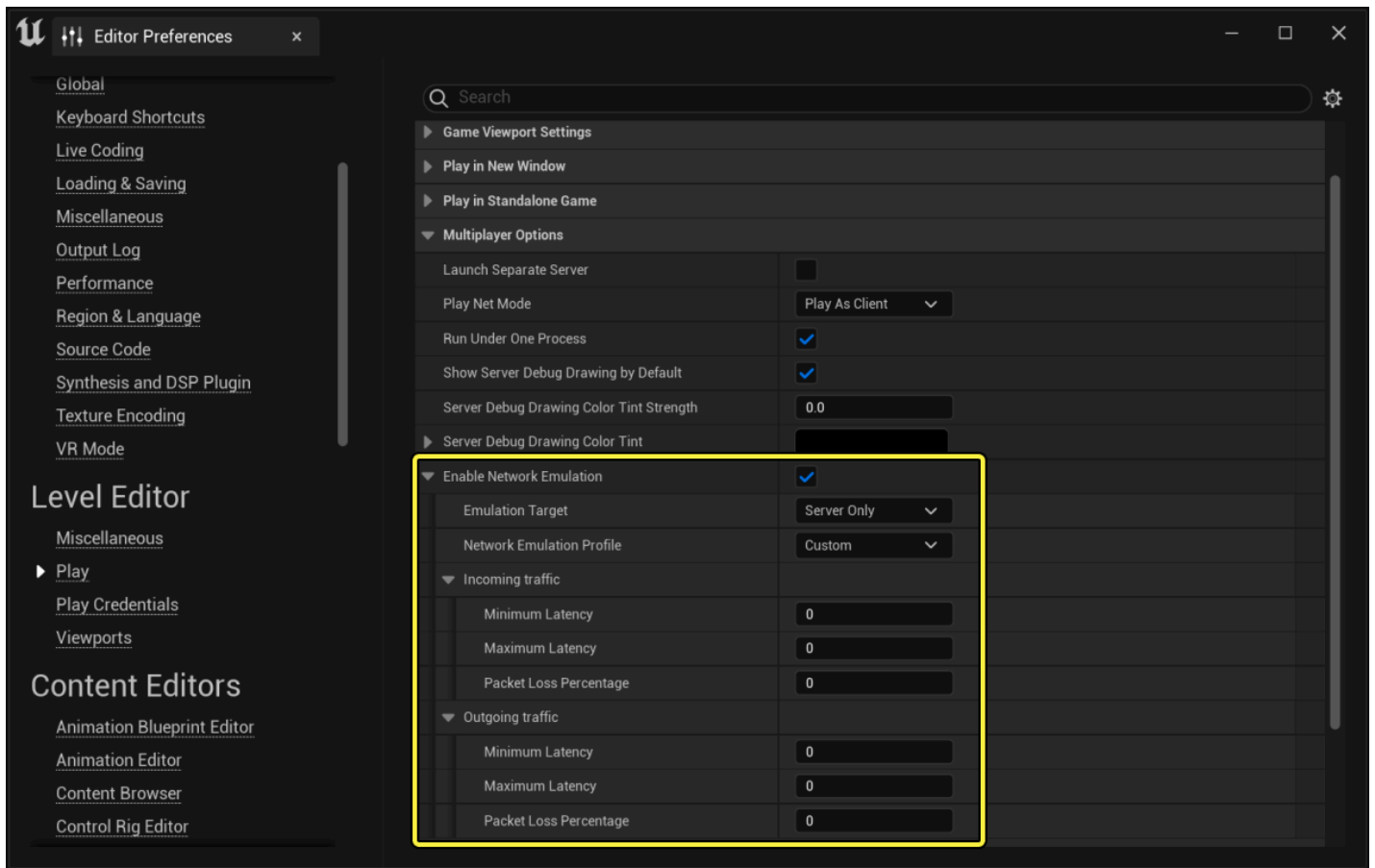
Testing multiplayer on a single machine requires you to run multiple instances of your game, with one acting as a server and the others acting as clients. Normally, this creates an environment with ideal conditions for your replicated data. Your replicated data will be passed through the appropriate systems, similar to a real networked environment, but there will be no lag or packet loss, which means you won't get an accurate idea of how it will behave when deployed to end-users. Similarly, LAN-based tests will not experience these conditions.

**Network Emulation** simulates lag and packet loss for servers and clients. This is crucial in identifying issues that may arise in networking environments. The Network Emulation settings are configurable in the **Unreal Editor**, command-line, console, and configuration files.

## Enable Network Emulation

### Unreal Editor

To use Network Emulation, from the **Menu Bar** navigate to **Edit > Editor Preferences > Level Editor > Play > Multiplayer Options** and enable the **Enable Network Emulation** checkbox. This enables the Network Emulation settings.



The editor includes pre-built **Network Emulation Profiles** including: **Custom**, **Average**, and **Bad**. These profiles use the provided fields to customize network emulation settings:

- **Incoming Traffic:** cause a delay or loss for receiving a packet
  - **Minimum Latency:** minimum amount of time lag in milliseconds (corresponds to `PktIncomingLagMin`)
  - **Maximum Latency:** maximum amount of time lag in milliseconds (corresponds to `PktIncomingLagMax`)
  - **Packet Loss Percentage:** chance that a packet is lost before being received (corresponds to `PktIncomingLoss`)
- **Outgoing Traffic:** cause a delay or loss for sending a packet
  - **Minimum Latency:** minimum amount of time lag in milliseconds (corresponds to `PktLagMin`)
  - **Maximum Latency:** maximum amount of time lag in milliseconds (corresponds to `PktLagMax`)
  - **Packet Loss Percentage:** chance that a packet is lost before being sent (corresponds to `PktLoss`)

# Command-Line

You can specify settings with a command-line argument using the syntax:


```
<SETTING_NAME>=<VALUE>
```

 Copy full snippet

# Console

You can specify settings from the console at runtime using the syntax:

```
NetEmulation.<SETTING_NAME> <VALUE>
```

 Copy full snippet

# Configuration Files

To set these values from your engine configuration files such as `DefaultEngine.ini`, add the `[PacketSimulationSettings]` section to your configuration file, followed by the `<SETTING_NAME>=<VALUE>` pairs you wish to specify:

```
1 [PacketSimulationSettings]
2 <SETTING_NAME>=<VALUE>
3 <SETTING_NAME>=<VALUE>
4 ...
```

 Copy full snippet

# Settings Reference

These settings provide you with a multitude of combinations to customize your network emulation:

| Setting                     | Description  | Value   |
|-----------------------------|--|---|
| <code>Off</code>            | Turn off network emulation   | Toggle command, no value required.  |
| <code>PktLoss</code>        | Specifies a percentage chance of an outbound packet being discarded to simulate packet loss.   | Clamped value between 0 (no packets dropped) and 100 (all packets dropped).       |
| <code>PktOrder</code>       | Changes the order of packets at random. This works by randomly selecting packets to be delayed until a subsequent call to <code>FlushNet</code> .  | 0 = False, Anything else = True   |
| <code>PktDup</code>         | Specifies a percentage chance of a sent outbound packet being duplicated.  | Clamped value between 0 (no packets duplicated) and 100 (all packets duplicated). |
| <code>PktLag</code>         | Delays the sending of a packet by the amount of time specified in milliseconds. <b>Note:</b> Cannot be used with <code>PktOrder</code> .   | Value in milliseconds.  |
| <code>PktLagVariance</code> | Causes packet lag to be a random variable instead of constant time in milliseconds. If set, lag becomes a random variable from <code>[PktLag - PktLagVariance, PktLag + PktLagVariance]</code> <b>Note:</b> Can only be enabled when <code>PktLag</code> is enabled. | Value is chosen at random from the range <code>[-Variance, Variance]</code> .     |
| <code>PktLagMin</code>      | Causes packets to have a minimum amount of lag.  | Value in milliseconds.  |
| <code>PktLagMax</code>      | Causes packets to have a maximum amount of lag.  | Value in milliseconds.  |

| Setting                          | Description   | Value   |
|----------------------------------|---|---|
| <code>PktIncomingLagMin</code>   | Causes packets to have a minimum amount of lag for the receiving end.   | Value in milliseconds.  |
| <code>PktIncomingLagMax</code>   | Causes packets to have a maximum amount of lag for the receiving end.   | Value in milliseconds.  |
| <code>PktIncomingLoss</code>     | Specifies a percentage chance of an inbound packet being discarded to simulate packet loss.   | Clamped value between 0 (no packets dropped) and 100 (all packets dropped). |
| <code>PktJitter</code>           | Causes sent packets to have a variable fluctuating latency by adding the value of <code>PktJitter</code> to range of possible packet lag times. | Value in milliseconds.  |
| <code>PktEmulationProfile</code> | Set a predefined emulation profile.   | Name of profile.  |

## Recommendations for Use

Multiplayer games running with perfect connectivity will not experience many bugs that only occur during poor connectivity. To ensure that you catch these bugs, you should perform local and LAN-based tests with extremely harsh conditions. For example:

- 500 round trip ping.
- 10% packet loss or higher.

**Epic Games** uses conditions like these to test cooperative and competitive gameplay. This makes it possible to catch numerous bugs and exploits that would otherwise go overlooked, and strongly incentivizes optimizing network performance.