MULTIPLAYER SERVICES

RESEARCH REPORT

Choosing the right netcode for your Unity multiplayer game

unity

There's no perfect, one-size-fits-all netcode solution for all the multiplayer games and experiences available. Each game must account for the network-related challenges that impact players' experience, such as latency, packet loss, and scene management, and games solve these challenges in a variety of ways. Finding the right solution depends on your game's genre, the scale of its players and networked objects, competitiveness, and other aspects, like how much control is needed over the networking layer.

We created this guide to help developers evaluate the options and decide which netcode solution (or combination of solutions) best suits a title's needs. Using data drawn from surveys and in-depth interviews with Unity users as well as in-house prototyping, we've rated the performance of some of the most popular netcode solutions in key areas.

Read on to learn about users' insights and experiences after launching multiplayer games with popular netcode solutions, and see our analysis of these solutions' strengths and weaknesses:

MLAPI (Best overall client-server solution)

DarkRift 2 (Best low-level solution)

Photon PUN (Simplest solution)

Photon Quantum v2 (Best deterministic rollback solution)

Mirror (Best UNet HLAPI evolution)

A high-level comparison table can be found here

Data sources

This document details the analysis and evaluation process that our team used to recommend third-party netcode solutions for multiplayer games built with Unity.

The team gathered and analyzed data from three sources:

- A survey of over 200 Unity users, in which we asked about their experiences with specific netcode frameworks
- 20 in-depth interviews with users actively shipping multiplayer games with Unity
- Learnings from prototypes we built with MLAPI, DarkRift 2, and Photon Quantum

We've scored and ranked the alternatives based on the following variables:

- Overview and overall recommendation
- Stability and support

- Ease of use
- Performance and latency
- Scalability
- Feature breadth
- Extensibility, debugging and contributing
- Console platform support
- Cost (and hidden costs)

Note: To get the most of this comprehensive overview, you should be familiar with key multiplayer and networking concepts. This talk highlights most of the essentials.

Best overall client-server solution: MLAPI

MLAPI (Mid-level API) is an open source networking library that offers a great breadth of mid-level features like NetworkedVars, scene management, remote procedure calls (RPCs), messaging, and more. This solution offers an abstraction layer to enable the swapping of different transports depending on the topology and platforms you plan to ship with. The solution assumes some form of client-server topology – either a dedicated game server (DGS, where clients connect and interact with the game on a central server) or a Listen Server (where one client hosts the server for the match).

By default, MLAPI assumes that you're using a Unity headless runtime for your server runtime, which enables you to write your physics or simulation just once, then use it for both the client and server.

Overall recommendation ★★★★★

MLAPI is the most well-rounded, high-quality solution we've found on the market today. It has great feature breadth at the mid-level, it's open source, its Discord community is active and very helpful, and the codebase is clean and well documented. As a result, it's quite feasible to start with MLAPI and extend or modify the solution to meet your needs. For most games shipping with Unity today, we would recommend trying MLAPI first.

Stability and support ★★★★★

MLAPI is a relatively new solution – its 1.0 version launched in April 2018. Since then, bug fixes and new features have been added and updated rapidly. The creator and others on the Discord channel respond to questions and issues quickly, and most developers who try MLAPI have positive things to say about its support and stability.

Ease of use ★★★★☆

MLAPI as a mid-level solution will already be much easier to use than something that has only low-level features. However, it's missing the Unity samples,

tutorials, and richer step-by-step guides that many users need to successfully get started with their first multiplayer game. It also does not yet offer a smooth migration path for existing games that already use UNet HLAPI (High Level API).

Performance and latency ★★★★★

MLAPI has been very careful to minimize garbage collection and allocations, optimize performance regularly, and keep its codebase clean. As a result, users report few or no challenges achieving their latency goals with MLAPI. Our own performance profiling confirmed the no-allocations claim. Users reported that fast-paced games have succeeded at hitting goals of ~50–100 ms latency.

Scalability ★★★☆☆

MLAPI unfortunately does not support multithreading by default, so it's not clear how far it will be able to scale in number of players or networked objects per session in shipping (non-sample) games. We know of at least one game that has succeeded at 64 players per session with MLAPI. Overall, it has good scalability as long as you avoid using convenience RPC APIs.

Feature breadth ★★★★★

In the mid-level, MLAPI is very feature rich. You can see the full list here, and its key features include:

- NetworkVars, SyncVars
- RPCs
- Scene management
- Messaging system
- Relay (which you can self-host)

At higher levels, however, lag compensation and delta compression are only partially supported, and many games may still need to create their own solutions for features like forward prediction. Unfortunately, we have not found any sufficiently stable and performant solutions on the market today with these high-level (HL) features.

Extensibility, debugging and contributing ★★★★★

MLAPI is a fully open source project, so it's very easy to debug, add project-specific features, and contribute if you are willing to share your work.

Console platform support ★★★☆☆

While MLAPI doesn't advertise console support out of the box, its abstraction layer for transports allows you to attach transport layers to work with various consoles, Steam, and so on. We've spoken to developers who have successfully shipped with a few different supported transports and were able to work with many different platforms.

Cost (and hidden costs)
Listen Server ★★★★★ or DGS ★★☆☆☆
This solution is fully open source and free, although it does not offer any hosted services (like dedicated server hosting or relay). The hidden costs lie in the services you need and how you choose to manage them.

If, for example, you want a completely free solution, MLAPI supports listen-server topology and makes it fairly easy to swap in the transport layers for each of the platforms that offer free relay services, like Steam, Xbox, and more. However, going this route will mean cross-play is not possible (i.e., Steam users cannot play with Xbox users). So, to offer that ability, you can still use MLAPI, but it will need to be connected to a paid relay and transport like Photon Realtime.

However, if you want cheat prevention and larger player scale, you can use MLAPI with a DGS topology, but then server hosting costs with Multiplay or another hosting service will be more expensive than using a relay.

Back to top ↑

Best low-level solution: DarkRift 2

DarkRift 2 is a fast and highly performant low-level networking solution – key features include bi-channel TCP and UDP, serialization, and customizable logging. The solution assumes a dedicated server topology, and it includes a multithreaded server runtime that can be extended with plug-ins. It is possible to use DarkRift with a Unity-based server and still use the plug-in system.

The entire solution is multithreaded by default, making it one of the most scalable solutions available. However, it does not use Unity's new C# Job System for its multithreading, so it does take additional learning and conversion to use the two together.

Overall recommendation ★★★★★
If you are a savvy developer who needs high performance and scalability, and you prefer to write your own mid- to high-level netcode on top of a stable baseline lower level, then DarkRift is a very strong solution that has proven production quality.

In addition, if you want a non-Unity server runtime to achieve greater efficiency, DarkRift also provides a highly performant solution for a server. Some users have managed to run it multi-tenanted when they wanted to optimize server footprint even further.

Stability and support ★★★★★
DarkRift 2 launched in July 2017 and has remained in development since then. It has a strong Discord community with many active developers and moderators

who are ready to help and answer questions at any time. Users have given a lot of positive feedback about the support for this solution.

Ease of use ★★★☆☆

Because DarkRift 2 is a lower-level solution, it will be inherently more difficult for new users to learn, and most users will need to be comfortable building additional netcode on top of DarkRift 2.

That said, there are tutorials and decent documentation, so users have been pleased with the ease of getting started with DarkRift 2.

Performance and latency ★★★★★

This solution was highly rated for performance, and our own investigation found that it has relatively low garbage collection and allocations, which can cause performance issues and latency spikes in more complex games. We hear that very fast-paced games have succeeded at hitting goals of ~50 ms latency.

Scalability ★★★★★

Of all the solutions, users rated DarkRift as the most scalable solution available today. Its multithreading by default gives it a distinct advantage over most other solutions.

Feature breadth ★★★☆☆

DarkRift 2 is a low-level solution, with a reasonable breadth of features. However, it doesn't offer any of the mid- or high-level features you may want, like NetworkVars, RPCs, and so on. And, as a low-level solution, it does not offer a reliable UDP channel out of the box, instead using TCP for reliable communication.

Extensibility, debugging and contributing ★★★★☆

You can purchase full source access for $100 on the Unity Asset Store, so it's possible to fully debug, extend, and modify the source to DarkRift 2. However, it is not an open source project at this time, and community members cannot contribute bug fixes back to the codebase.

Console platform support ★★☆☆☆

As a low-level solution, we'd hope to see clear methods for supporting console platforms, but this solution unfortunately does not appear to be ready to pass console certification out of the box. It should be feasible to plug in a different transport that is capable of supporting consoles, but we have not spoken with any users who have successfully done so.

Cost (and hidden costs) ★★★☆☆

DarkRift 2 is free for a subset of features and no source access, and it costs a one-time fee of $100 to gain access to the code and full features. Its hidden costs are dedicated server hosting (not provided by the solution), which can be expensive depending on the provider or how you host it.

Back to top ↑

Simplest solution: Photon PUN

Photon PUN is essentially a mesh topology solution (a.k.a. direct P2P), in which each client synchronizes everyone else's data.

It includes a relay service to avoid NAT traversal issues, referring to the relay as a "Master Server." This term can be misleading, since the relay/server is not doing any synchronization logic. It has the ability to run simple logic on the Master Server via plug-ins to check and modify messages as these pass through, which could be used for cheat detection (but not cheat prevention as with a DGS).

The benefit of a mesh topology is that there's no single host/server to manage. However, the downside is that every client has to manage the synchronization logic of every other player, which limits the scale of the game, and there is no server authority to prevent cheating. This means that this solution has some inherent risks, and adopting this topology can make it very hard to transition later to a server-authoritative model.

Overall recommendation ★★★☆☆

Users who love PUN appreciate its simplicity and often describe it as dead simple. If you are making a 2–4 player game that is cooperative or very casual, this solution may be sufficient for your needs. Hobbyist and student projects might also appreciate the ease of use and rapid onboarding. However, choosing this solution will make it very difficult to later convert to a client-server topology, and your game will always be locked at a small scale (maximum 8 players).

Stability and support ★★★★☆

Exit Games (the company behind all Photon products) has existed for some time, and PUN is a fairly stable solution with minimal bugs or issues. However, users cite challenges getting support for questions and issues, and there doesn't seem to be as strong a community as MLAPI or DarkRift has fostered.

Ease of use ★★★★★

Photon PUN is regularly called out as the simplest solution to get started, boasting a strong Unity SDK, tools, and very simple interfaces.

Performance and latency ★★★☆☆

Users say latency can sometimes be problematic with PUN, so it may not be viable for very fast-paced games. However, if your game can tolerate moderate latency of 150–200 ms, this solution may still be sufficient.

Scalability ★☆☆☆☆

A direct P2P topology struggles with scale in any implementation, and it depends on the capabilities of the client hardware and what you are willing to sacrifice (in terms of compute) to handle more synchronization logic on each client. In general, most users who find success with mesh topology games have a maximum of 4–8 players per session, and that's true for PUN users as well.

Feature breadth ★★★★☆

PUN is essentially a full-featured mid-level solution for mesh topology games that includes key features such as:

- RPCs
- Serialization
- Hosted relay and simple matchmaking

However, higher-level features like prediction and delta compression are missing.

Extensibility, debugging and contributing ★☆☆☆☆

Photon PUN is closed source, so it's impossible to step through to debug code. Extension beyond their plug-in system is also not possible, as is rewriting or optimizing directly. Since this is not an open source community project, there is no way to contribute back to Photon PUN.

Console platform support ★★★★★

Photon is supported on most major platforms, including being a supported middleware provider for major consoles.

Cost (and hidden costs) ★★★☆☆

Photon PUN is free to try up to 20 concurrent users, after which you can pay a one-time fee of $95 for 100 concurrent users per month, and $0.29 per user beyond that. While this is not free, it is fairly low cost relative to DGS solutions.

It's important to note that this solution is closed source and requires you use their relay for every player. The upside is that cross-play is feasible across platforms. The downside is that you will never be able to leverage the free relays on many platforms (like Steam or Xbox) to reduce costs.

Back to top ↑

Best deterministic rollback solution: Photon Quantum v2

Photon Quantum is essentially the simulation portion of a game engine focused on the core functionality required for deterministic rollback games. Quantum v2 uses an Entity Component System (ECS) and multithreading systems that are not the same as Unity's ECS and Job System, and to use Quantum you must be willing to use their fixed-point math, physics, pathfinding, and other simulation libraries instead of Unity libraries. The solution relies on Unity for the presentation and Editor, however, and there can be some challenges converting between the two.

In deterministic rollback solutions, only user inputs are sent (i.e., pressed x instead of shot in y direction), and every other user accepts those inputs

and runs their version of the simulation locally. On each client, it runs logic to predict and visualize where other players may be while waiting for new inputs, then it rolls back the simulation to replay and correct the positions once inputs arrive. This gives a smoother feel than pure lockstep. Because of determinism in simulation, you can ensure the simulations stay in sync on each device, and even if a user tries to cheat and boost their character, this will impact only their local simulation but not that of others. Quantum also allows some ability to run the same simulation on a server to act as a referee.

The biggest drawback to this system is the heavy reliance on client hardware to handle all of the synchronization, prediction, and rollback computation for every other player in the entire simulation. This means that your scale will be limited by the power of the game clients. So, realistically, you can likely run only small-scale games on mobile and larger ones (they claim it's proven up to 32 players) on heartier PCs or consoles. Fixed-point math, relatively simple physics, and 2D-only navmesh and pathfinding may also deter simulation-heavy or complex games where precision matters (like FPS) from using this solution.

Overall recommendation ★★★★☆

This solution seems solid for games that do best with deterministic rollback solutions with 2D-ish views (including top-down) – typically RTS, MOBA, Clash Royale, or fighting games. The pricing is likely less than a DGS topology for the same genres and ultimately may be worth it for these kinds of games, though it is considerably more expensive than other solutions for simpler games.

Stability and support ★★★★☆

Users of Photon Quantum v2 state that even though there are still some bugs, the team has been quick to resolve issues, and there seems to be ongoing growth. It's generally perceived as a more stable alternative to Unity's own deterministic/ECS libraries for the time being. Unfortunately, Photon solutions lack the thriving communities of other offerings.

Ease of use ★★★★☆

In our own prototyping, we were able to achieve a fully functioning prototype RTS game within ~1 week of person-hours, and it has been described as quite easy to use overall. The fact that you only send inputs makes the networking part of the solution easier to achieve, although the ECS simulation and synchronization logic reintroduce some complexity.

Performance and latency ★★★★★

Quantum uses the same core infrastructure as PUN – i.e., relay and transport, and we know that there are people who have had latency issues with the Photon cloud when using Bolt/PUN, so that may also apply to Quantum.

That said, the perceived latency with Quantum is lower than with a standard P2P architecture because Quantum's prediction model predicts the entire game state.

In addition, Quantum has very minimal garbage collection and allocations. The performance of the system is generally good because it uses a memory-aligned ECS for calculating the game state, but compared to DGS solutions, Quantum will often perform worse because it has to simulate everything on the client.

This solution is likely sufficient for reasonably fast-paced games.

Scalability ★★☆☆☆

As mentioned above, scalability is one of the biggest challenges with this topology, especially if you want the game to work on low-powered mobile devices. In the showcase of games we've seen for Quantum, most stay within 4–6 players per session, although it seems that at least one has been able to hit as high as 32 players per session.

Feature breadth ★★★★☆

Quantum is a full-featured solution for deterministic rollback games; most critical features exist today.

However, reliance on fixed-point math, relatively simple physics, and 2D-only navmesh means they require more features before they're ready to support additional genres beyond typical RTS, MOBA, Clash Royale, or fighting games.

Extensibility, debugging and contributing ★★☆☆☆

Photon solutions are not full source, and therefore do not enable you to step through to identify and fix bugs yourself. Similarly, since it's not an open source project, users cannot contribute fixes back to the repository. However, the Quantum solution enables developers to tailor game code to their game reasonably well, and some users cite that the black box was at least a predictable black box, so they could extend it fairly easily.

Console platform support ★★★★★

All Photon solutions depend on real-time transport, relay, and matchmaking, which means console and cross-play are feasible from the outset. It's worth noting that Quantum cannot support WebGL.

Cost (and hidden costs) ★★★☆☆

Quantum costs $1000 per month for five seats, and $0.50 per concurrent user. This means that for larger teams, the development costs could get quite expensive. However, the operating costs are likely lower than most DGS hosting fees for operating the same kinds of genres, so this still may be a reasonable path for some games.

Back to top ↑

Best UNet HLAPI evolution: Mirror

Mirror is a fork of the now deprecated UNet HLAPI that has been adopted and supported by the community. Like HLAPI, it focuses on supporting client-server topology games and offers a number of mid-level features like SyncVars.

Unfortunately, the solution's core architecture is based on UNet, which had fundamental challenges with achieving high performance and scalability. Primarily for this reason, this solution may not be sufficient for games with high scale needs.

Overall recommendation ★★★☆☆

Mirror is easy-to use and has a great community. It's a particularly solid solution if you have an existing game that is using UNet HLAPI successfully, and you want to move to something that is better supported.

Stability and support ★★★★★

Of all the solutions, Mirror currently has the largest and most active Discord community. This is one of the greatest benefits of Mirror, and many users note how great this community is. In addition, since forking from HLAPI, many bugs have been fixed, so the solution is much more stable than it was previously.

Ease of use ★★★★☆

Like HLAPI, Mirror is relatively easy to use, with a good number of samples and improved documentation from the past. Overall, many users find it easy to get started with Mirror.

Performance and latency ★★★☆☆

Unfortunately, HLAPI struggled with a monolithic design that made it hard to optimize performance. Mirror struggles with the same core design, and it uses LINQ libraries heavily to reduce codebase size. In our profiling, the memory allocations and garbage collection spikes could get pretty rough as a result, leading to spikes in performance issues.

Scalability ★★★☆☆

Like HLAPI, there are inherent scale challenges in Mirror. We know of a few games that heavily modified HLAPI to reach 64 players per session, but without significant changes to HLAPI, we haven't identified many shipped games have succeeded with more than 32, and the vast majority are closer to ~4 players per session.

Feature breadth ★★★★☆

As a mid-level set of features, Mirror is reasonably complete, with key features that include:

- SyncVars
- RPCs
- Multiple transports supported

Additionally, high-level features like prediction or delta compression are also missing.

Extensibility, debugging and contributing ★★★★☆

Mirror is a fully open source project, so it's possible to debug it directly and contribute fixes back to the community. However, the core design of the code is not ideal for modification and extension (i.e., it's monolithic and has heavy LINQ library usage).

Console platform support ★★★☆☆

Mirror on its own does not support consoles. However, it supports a wide variety of transports that may make it feasible to support console platforms.

Cost (and hidden costs)
Listen Server ★★★★★ **or DGS** ★★☆☆☆

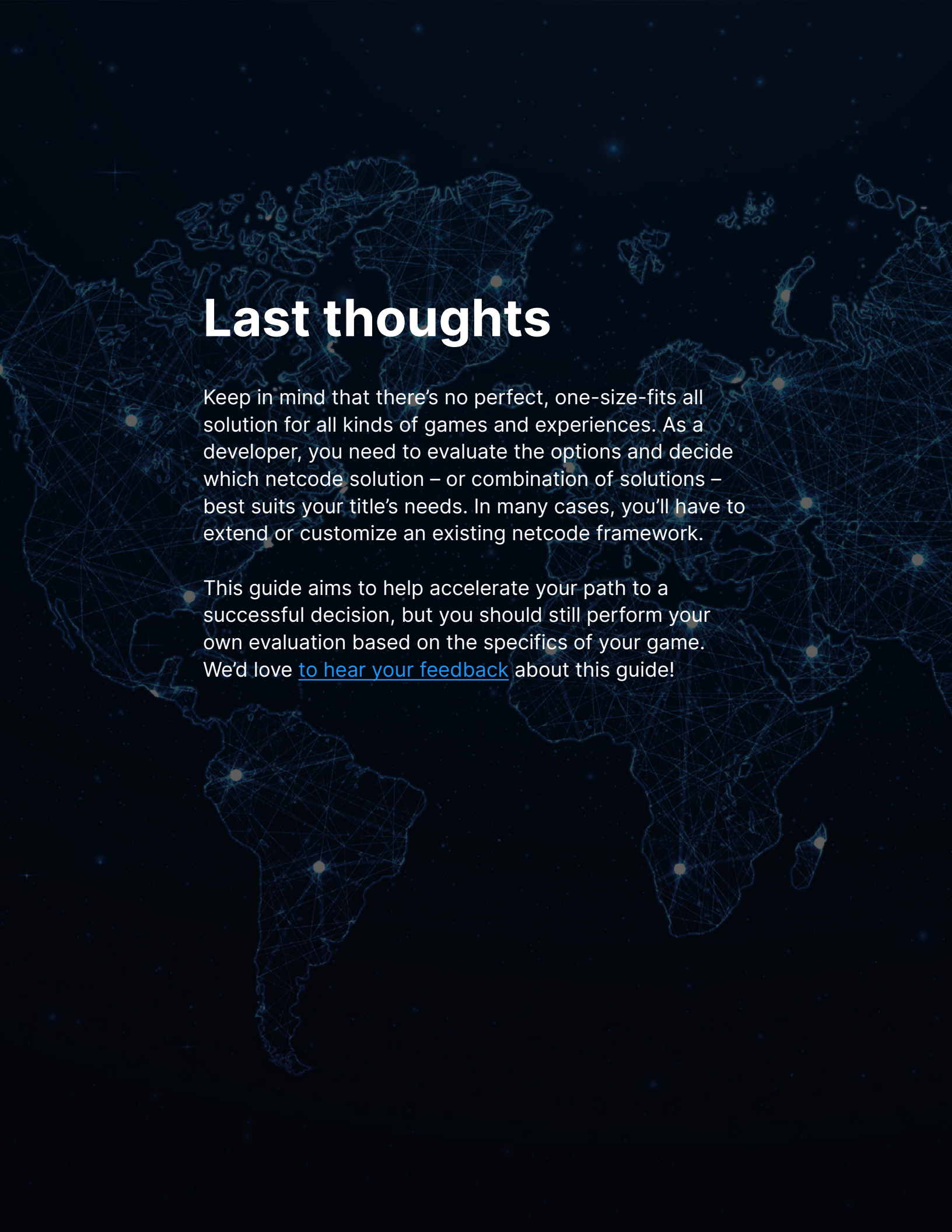
Mirror is free, though like MLAPI, hidden costs lie in the services you need and how you choose to manage them. It's potentially free if you're using a listen-server topology or fairly expensive with DGS.

Back to top ↑

Comparison table

	Stability/ support	Ease-of-use	Perfor-mance	Scalability	Feature breadth	Cost*	Recommended for
Best overall **MLAPI**	★★★★★	★★★★☆	★★★★★	★★★☆☆	★★★★★	Free	Most client-server games for up to ~64 players that want a stable breadth of mid-level features
Best low-level **DarkRift 2**	★★★★★	★★★☆☆	★★★★★	★★★★★	★★★☆☆	$100 for source	Games with high perf/ scale requirements that want to build on a stable LL layer
Simplest **Photon PUN**	★★★★☆	★★★★★	★★★☆☆	★☆☆☆☆	★★★★☆	$0.30/PCU	Simple and small (2–8 players) mesh-topology games
Best deterministic **Photon Quantum 2.0**	★★★★☆	★★★★☆	★★★★★	★★☆☆☆	★★★★☆	$1000/mo + $0.50/PCU	Games desiring deterministic roll-back, like MOBA games, for up to 32 players
Best HLAPI evolution **Mirror**	★★★★★	★★★★★	★★★★☆	★★★★☆	★★★★☆	Free	Client-server games already using UNet HLAPI that want a more supported solution

* Note that Photon pricing provides access to the networking libraries and services, whereas other solutions are standalone networking libraries, and the cost of services is separate.



Last thoughts

Keep in mind that there's no perfect, one-size-fits all solution for all kinds of games and experiences. As a developer, you need to evaluate the options and decide which netcode solution – or combination of solutions – best suits your title's needs. In many cases, you'll have to extend or customize an existing netcode framework.

This guide aims to help accelerate your path to a successful decision, but you should still perform your own evaluation based on the specifics of your game. We'd love to hear your feedback about this guide!


unity