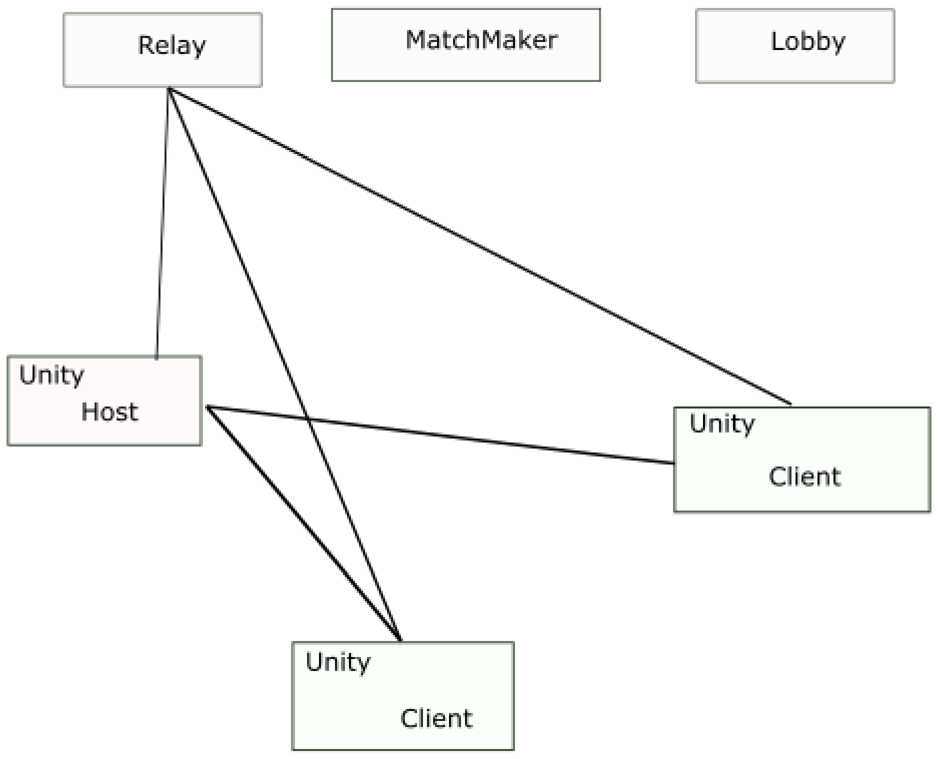
UNET

개요

Unity Multiplayer Networking. P2P로 클라이언트 중 하나가 호스트 (Dedicated Server)가 되는 방식이다. 매치 메이킹 등 로비 기능을 서버에서 제공하고 게임 플레이는 P2P 로 진행하는 방식이다.

https://bitbucket.org/Unity-Technologies/networking

비트버켓에서 오픈 소스로 진행되는 프로젝트이다. 따라서, 세부 구현을 파악할 수 있다.



사용 방법

MatchMaker 서버를 사용하여 게임에 들어오는 클라이언트를 식별한다. 이후 P2P로 통신이 진행된다. P2P 연결이 불가능할 경우 릴레이 서버를 사용한다. 매치메이커는 자체적으로 구현해도 된다.

C# 변수에 속성을 주는 방법이 몇 가지 이슈가 있어 보인다. 사용자가 리포트 한 바로는 오브젝트 인스턴스 모두에서 동작하는 경우가 있고 어떤 경우는 동작하지 않는 경우가 있으며 이동 동기화도 기능이 좋지 않다는 평가다. 따라서, 트래픽을 측정하면서 개발을 진행해야 한다.

이슈들

NAT 홀 펀칭

UDP 기반인데 NAT 홀 펀칭 기능이 없다. 우회하는 방법은 RakNet의 기능을 사용하는 방법이 있다. 자체적인 릴레이 서버가 있으나 느리다는 평가다.

https://forum.unity.com/threads/unet-relay-server-teribble-latency.457408/

포톤 썬더 (Photon Thunder)의 릴레이 기능이 더 빠르다는 리포트가 위의 문서에 있다.

동기화 등

아래 사용자 평가에 있는 항목들은 모두 꼼꼼히 체크해서 구현해야 한다.

테스트

구현이 되면 꼼꼼한 테스트가 필요하다.

- 방화벽 / NAT 환경에서 플레이 진행
- 동기화 정확성
- 트래픽 측정
- 릴레이 시 동기화

부하 테스트는 P2P로 진행되므로 크게 의미는 없다. 매치메이커 등의 UNET 자체 기능을 사용할 경우 적정한 부하에 대해 조사하고 진행한다. 확장이 가능한 구조인 지 확인한다.

사용자 평가

wireshark를 쓸 수 있는 사람의 평가니 믿을 만 하다.

Oh... UNET is silly. Much of my development time was spent working around weird "gotchas". I spent several hundred hours learning (read: fighting) UNET. Here's a list of things that I wish I would've known.

What I hate(d) about UNET:

- Changing simple, built-in, settings on the server, before instantiating objects , does not sync to clients *for the Network Lobby Manager*. The Unity IRC at the time told me to never use the UNET managers; always write your own. It was good advice.
- SyncVars do not sync *most of the time* on *instantiated network behaviors* that are attached to an existing object after the network object is spawned. This took me a long time to fully track down, since all the other NetworkBehaviour methods seem to work on instantiated behaviors (Commands, RPC, isServer, etc).

- ClientRPCs are called on ALL instances of an object, not the instance that the command was called on. UNET (somewhat) recently added TargetRPC that addresses this issue.
- Network Behaviors will sometimes stick around in the editor, even though no network connection exists, but will never stick around in the Client. I've lost sleep over this issue. In fact, there are a lot of behavior differences between the editor and builds, specifically with UNET, but I can't remember them all right now.
- **NO BUILT-IN NAT PUNCHTHROUGH**. Yes, really. I had to write my own. However, a few months ago I saw a NAT punchthrough asset on the asset store, so this might only cost $40 to fix.
- SyncLists sometimes just *didn't work*. I ended up writing my own message to sync a list of integers reliably, and haven't used SyncLists since. They might be fixed now; I did see an open bug about them over a year ago.
- Unity's matchmaking servers *were* expensive compared to Photons. Though, writing your own is the way to go here.
- NetworkTransform's interpolation simply didn't work for kinematic objects. It might work now, haven't tried in awhile. Though, writing your own interpolation with position and rotation SyncVars works (unless the NetworkBehaviour was added to the object later).
- Photon simply has more development behind it. The documentation is better, and there are fewer unanswered questions on the forum. So often I'd encounter an issue with UNET, find someone with the exact same issues, but zero answers or comments. Likewise, I'd encounter UNET documentation that was just flat-out incorrect.
- Lock-step was much trickier in UNET than with Raknet. If you're not doing an RTS, then this might not matter to you.
- Unity does a lot of magic behind the scenes. Wireshark was necessary to figure out exactly what this magic is.

UNET is weird. They try to simplify networking for those without networking experience, and I think it just comes across as obnoxious. However, once I learned all the weird aspects of UNET, it seems to work pretty well.

I wish I would have used Photon hundreds of hours ago, but now I'm okay with UNET, and I think UNET will get much better over time.