Photon Fusion

nguyenkimlong@savameta.com

Overview

- Hỗ trợ các phương pháp giảm trễ qua mạng.
- Hỗ trợ đồng bộ hóa trạng thái qua mạng.
- Hỗ trợ nén dữ liệu trạng thái qua mạng.
- Hỗ trợ hẹn giờ qua mạng.
- Hỗ trợ RPCs.
- Hỗ trợ 3 chế độ chơi: Shared, Hosted, Server.
- Hiệu năng vượt trội so với Mirror và MLAPI.
- Tích hợp tự nhiên vào Unity.
- Và còn nhiều tính năng khác

Fast-Paced Multiplayer

Có thể tóm tắt trong 3 câu sau:

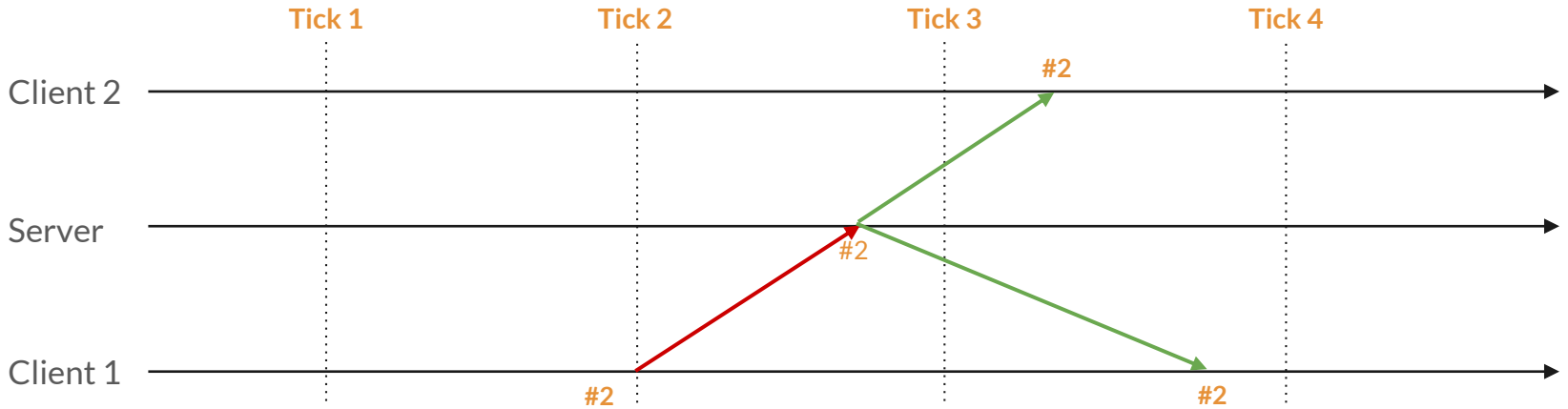
- Dự đoán máy mình
- Nội suy máy khác
- Bù trễ máy chủ

Photon Fusion hỗ trợ giảm trễ dựa trên Tick. Vậy Tick là gì?

Tick

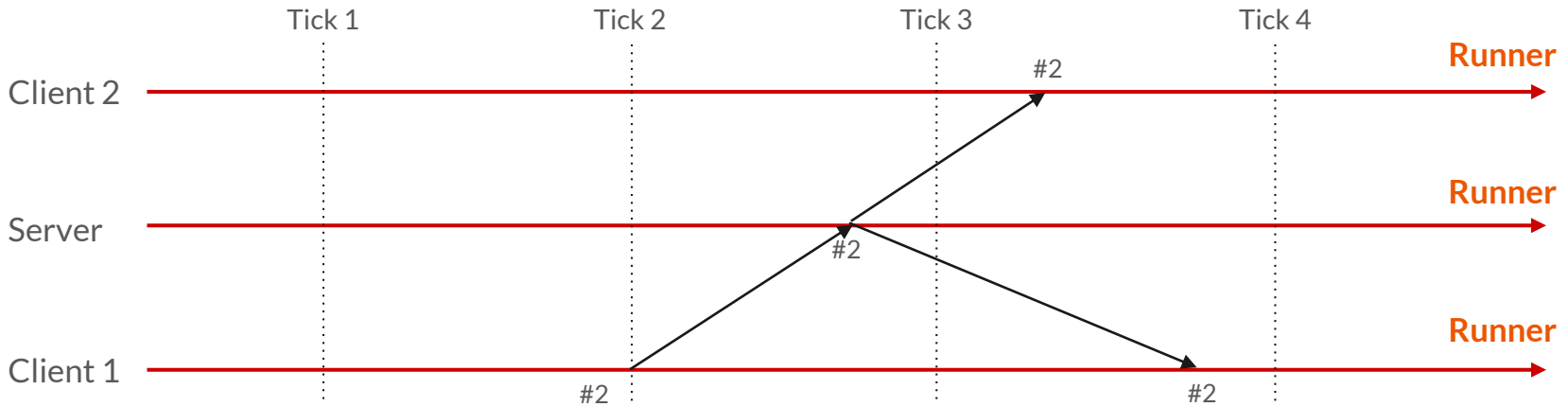
Tick giúp ta nhớ được vị trí của gói tin mà không cần tới thời gian cụ thể. VD: Client 1 gửi gói tin #2 tới Client 2. Client 2 tự phải biết cách chia 1s thành nhiều tick, sau đó ghi đè thông tin của gói tin ở vị trí tick số 2.

- Để quản lý tốt các gói tin gửi tới máy chủ, Photon Fusion dùng Tick làm đơn vị.
- Tick là tần số gửi các gói tin. VD: Tick = 60 => có 60 gói tin được gửi tới máy chủ trong 1s.



Runner

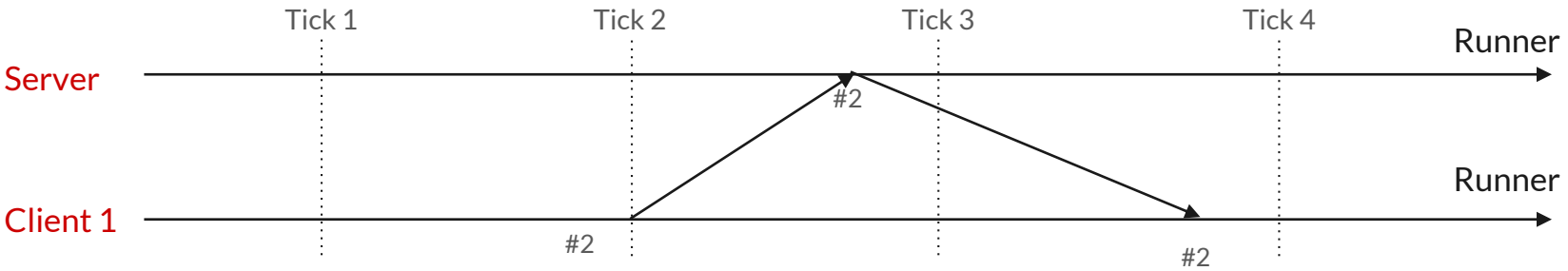
- Để cho Developer can thiệp vào Tick, Photon Fusion cung cấp đối tượng có tên Runner.



Runner

Hỗ trợ kiểm tra chế độ:

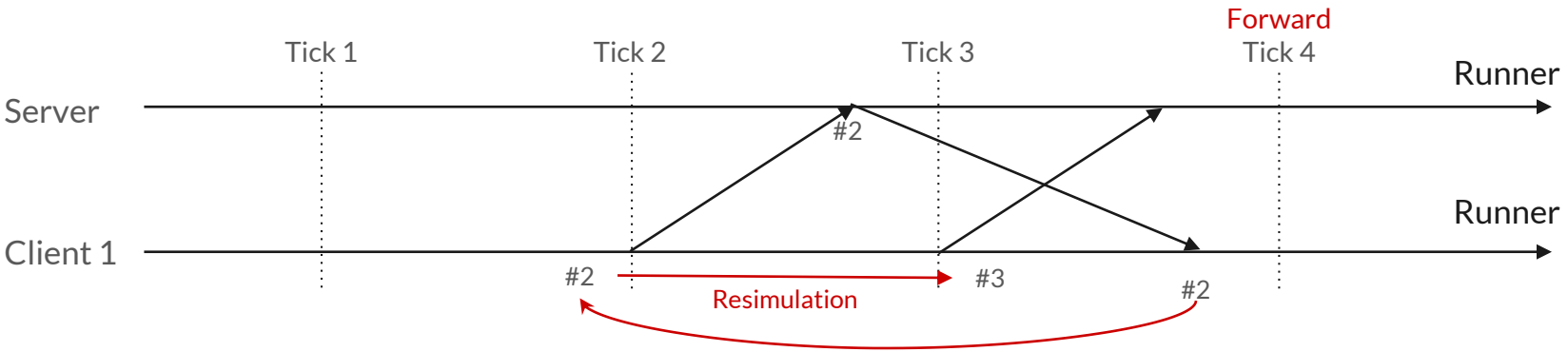
- IsServer: Máy chủ.
- IsClient: Máy khách.



Runner

Hỗ trợ kiểm tra quá trình:

- IsResimulation: Là quá trình cập nhật lại trạng thái.
- IsForward: Là quá trình thực thi hiện tại.

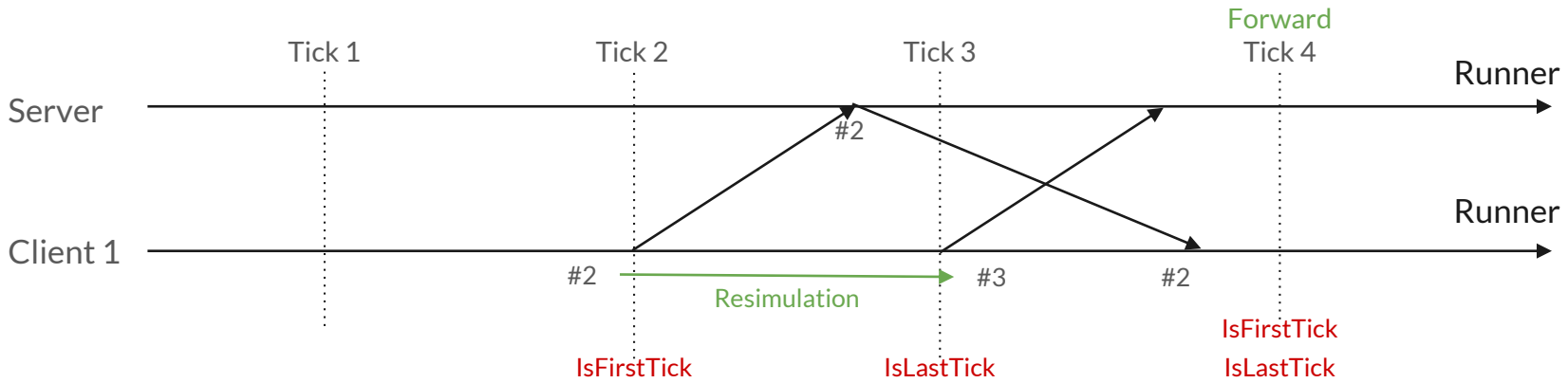


Runner

Photon Fusion có lưu ý rằng trong một số trường hợp Forward sẽ có thêm 1-2 tick. VD: Khi load scene, mọi thứ sẽ ngừng update để tăng tốc độ load scene, nhưng Tick vẫn sẽ tiến về phía trước.

Hỗ trợ kiểm tra Tick:

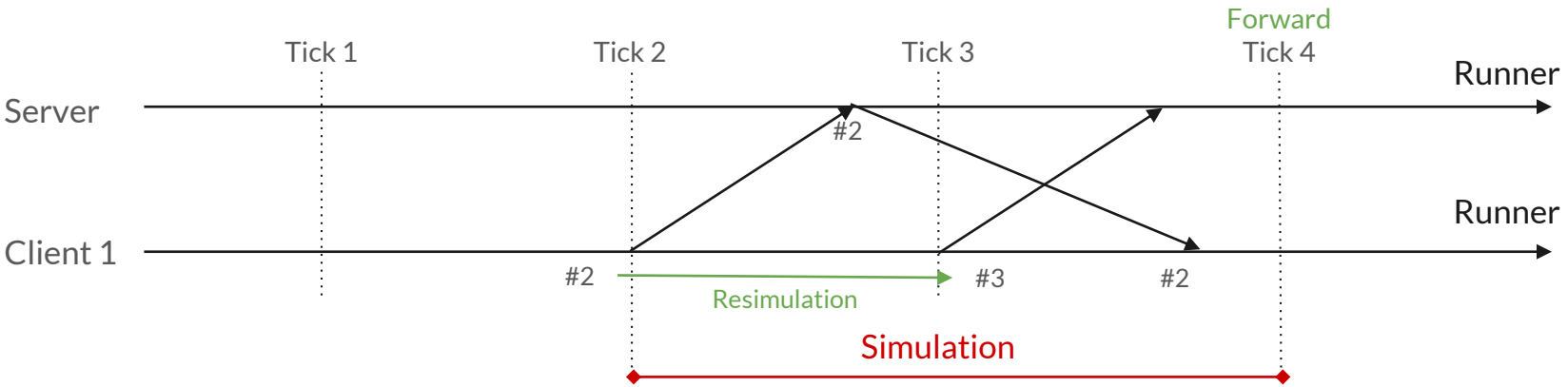
- IsFirstTick: Tick đầu của quá trình.
- IsLastTick: Tick cuối của quá trình.



Simulation

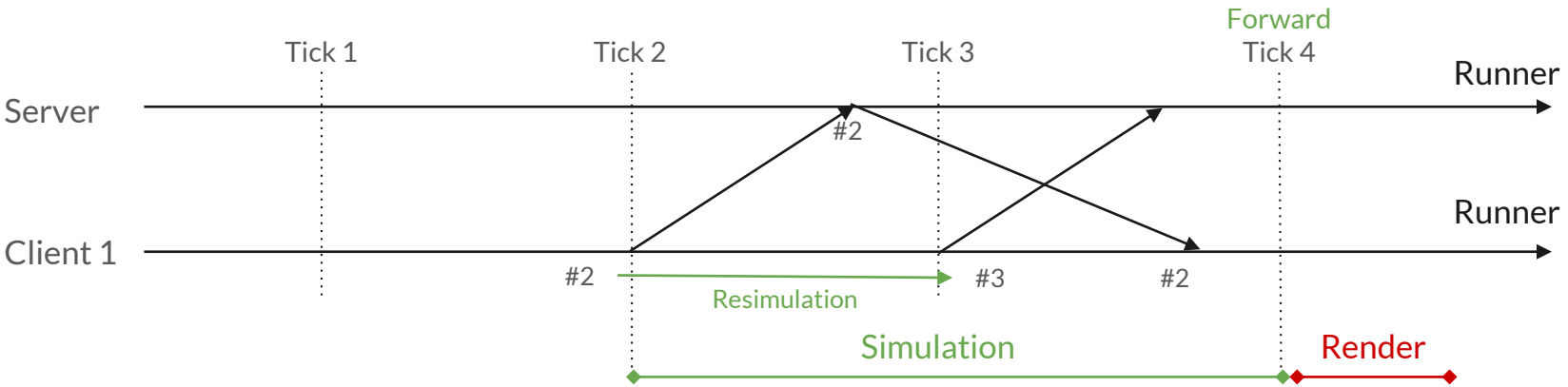
Simulation = Resimulation + Forward. Simulation hỗ trợ kiểm tra thẩm quyền:

- HasStateAuthority: Có quyền can thiệp vào trạng thái của đối tượng.
- HasInputAuthority: Có quyền can thiệp vào đầu vào của đối tượng.



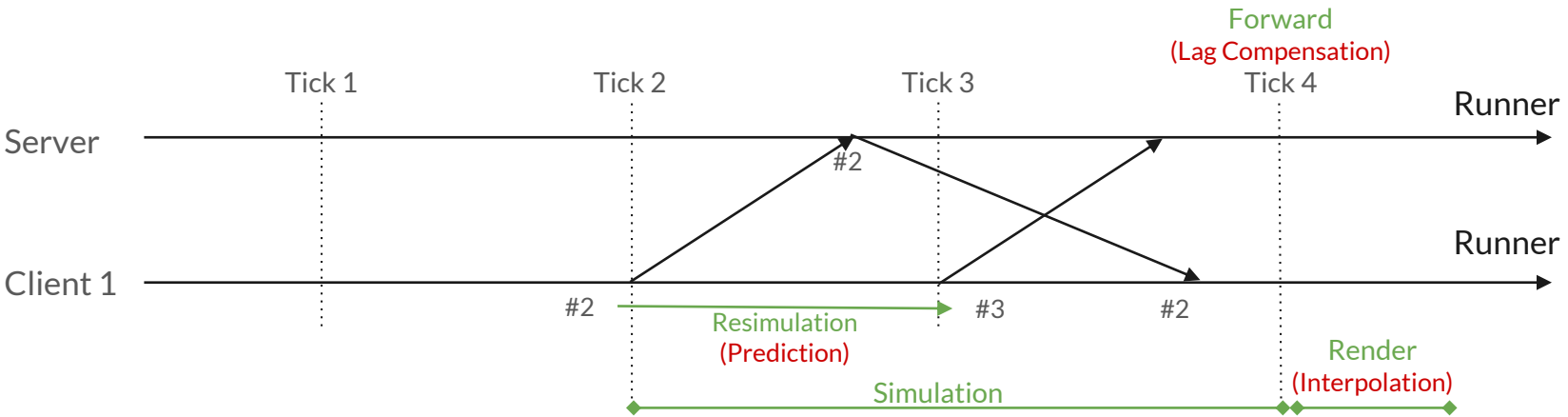
Render

- Là toàn bộ những gì không nằm trong Simulation.
- Hỗ trợ kết xuất ra hình ảnh, hoạt ảnh và VFX.



Cách áp dụng các phương pháp giảm trễ

- **Prediction** code trong Resimulation.
- **Interpolation** code trong Render.
- **Lag Compensation** code trong Forward.



Prediction

- **Prediction** thường liên quan tới các đối tượng có InputAuthority.
- Khi có InputAuthority thì ta có thể di chuyển đối tượng luôn mà không cần chờ xác nhận của máy chủ.

```
if (GetInput(out NetworkInputData data))
{
    _cc.Move(5 * data.direction * Runner.DeltaTime);
}
```

Interpolation

- **Interpolation** dùng kiểu dữ liệu **Interpolator** để lưu giá trị cần nội suy.

```
[Networked] public float Rotation { get; set; }
public Interpolator<float> Interpolator;

public override void Spawned() {
    Interpolator = GetInterpolator<float>(nameof(Rotation));
}

public override void Render() {
    transform.rotation = Quaternion.Euler(0, 0, Interpolator.Value);
}
```

Lag Compensation

- **Lag Compensation** chứa trong Runner và hỗ trợ Raycast theo LayerMask

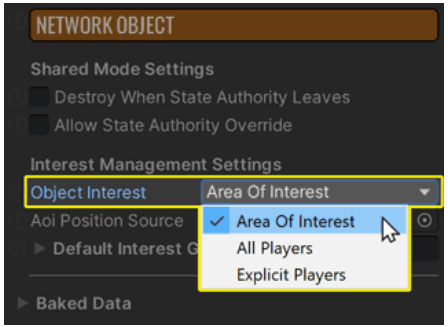
```
public override void FixedUpdateNetwork()
{
      var origin = transform.position;
      var direction = transform.forward;
      var length = 10f;
      var playerRef = Object.InputAuthority;
      var hits = new List<LagCompensatedHit>();
      var layerMask = -1;
      Runner.LagCompensation.RaycastAll(origin, direction, length, playerRef, hits, layerMask);
}
```

State Transfer

- Là cách mà Photon Fusion đóng gói các trạng thái thành tập tin.

- Có 2 chế độ:
  - Delta-Compressed (DC): Nén toàn bộ trạng thái rồi gửi đi.
  - Eventual Consistency (EC): Nén một phần trạng thái rồi gửi đi.

- EC được sử dụng kết hợp với **Interest Management** để quản lý những đối tượng mà người chơi quan tâm, những đối tượng này có 3 chế độ:
  - Area Of Interest: Tạo ra 1 vùng không gian hình vuông theo chiều ngang, khi người chơi đi vào thì mới sử dụng EC.
  - All Players: Áp dụng EC với tất cả người chơi.
  - Explicit Players: Chỉ áp dụng EC với người chơi được chỉ định.



TickTimer

Photon Fusion hỗ trợ hẹn giờ bằng đối tượng TickTimer.

Theo ví dụ bên, khi được tạo ra, Ball sẽ lưu lại thời gian tồn tại của nó là 5s.

Trong vòng cập nhật của Photon, nếu thời gian tồn tại là hết hạn, Ball sẽ bị xóa bỏ.

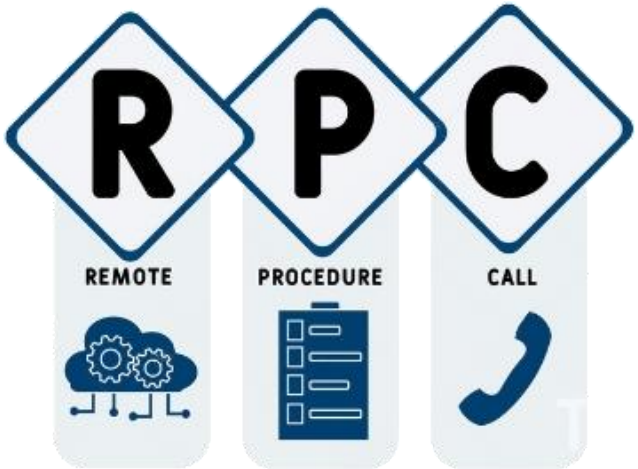
```
public class Ball : NetworkBehaviour
{
    [Networked] private TickTimer life { get; set; }

    public override void Spawned() {
        life = TickTimer.CreateFromSeconds(Runner, 5.0f);
    }

    public override void FixedUpdateNetwork() {
        if (life.Expired(Runner))
            Runner.Despawn(Object);
        else
            transform.position += 5 * transform.forward *
Runner.DeltaTime;
    }
}
```

RPCs

- RPCs = Remote Procedure Calls.
- Là phương pháp để chia sẻ sự kiện trò chơi một cách chính xác.
- Lý do là vì cách truyền trạng thái có thể có sai số hoặc mất dữ liệu, nên trong một số trường hợp cần dữ liệu chuẩn xác và không liên tục, RPCs sẽ phát huy tác dụng.
- VD: Khi người chơi thực hiện một tương tác phức tạp hiếm gặp với một đối tượng không có InputAuthority, chẳng hạn sử dụng một khóa cụ thể từ kho đồ của họ để mở một cánh cửa bị khóa, nếu mở sai thì cửa bị khóa vĩnh viễn.



RPCs

Khai báo một hàm với tiền tố là **RPC_** với việc chỉ rõ rằng RPCs sẽ đc gọi từ đâu (Source) tới đâu (Target).
Ngoài ra, cuối hàm có thể chứa RpcInfo, cung cấp một số thông tin như: Tick, Source, Channel, IsInvokeLocal.

```
public class Player : NetworkBehaviour {
    [Networked] public string playerName { get; set; }

    [Rpc(RpcSources.InputAuthority, RpcTargets.StateAuthority)]
    public void RPC_Configure(string name, RpcInfo info = default){
        playerName = name;
    }
 }
```

Hack

- Vấn đề gian lận sẽ làm cho Game của bạn kém hấp dẫn do mất sự công bằng.
- Photon Fusion cũng không thể tránh khỏi việc bị hack, sau đây là một số trường hợp có thể bị hacker khai thác:
    - Hack dựa vào logic
    - Hack dựa vào authority
    - Hack dựa vào RPCs

Hack dựa vào logic

VD1: Tốc độ click chuột của mọi người rơi vào khoảng 20ms. Nhưng nếu một hacker dùng phần mềm click chuột liên tục để gửi input lên server thì sao?

VD2: Tiền trong Game sẽ tăng giảm tùy thuộc cách chúng ta mua/bán hàng hóa. Nhưng nếu Hacker mua/bán liên tục thì sao?

Đối với VD1, ta có thể dùng TickTimer để hẹn giờ cho click, khi click hết hạn thì mới được click tiếp.

Đối với VD2, ta có thể giới hạn số lượt giao dịch của 1 người trong 1 khoảng thời gian, hoặc trong một phiên giao dịch.

Hack dựa vào authority

VD1: Hacker vào nhà của một người mà không cần chìa khóa.

VD2: Hacker lấy được súng của người dùng khác mặc dù họ vẫn đang cầm.

Đối với cả 2 VD trên, ta đều cần kiểm tra thẩm quyền trước bằng lệnh HasStateAuthority và HasInputAuthority.

Hack dựa vào RPCs

VD1: Giả sử ta có RPC để mở khóa vật phẩm, hacker dùng RPC đó để mở khóa mà không cần thực thi hết nhiệm vụ.

VD2: Giả sử ta có RPC để giết một người dựa vào tên của họ, hacker có thể truyền tên người khác vào để giết bất kỳ ai.

Đối với cả 2 VD trên, nên check điều kiện trong RPC một cách chặt chẽ và hết sức chú ý khi RPC có thể làm một thao tác ảnh hưởng hệ thống.