HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA ĐA PHƯƠNG TIỆN

---------o0o----------

BÁO CÁO

LẬP TRÌNH GAME NÂNG CAO

Chủ đề: Các giao thức kết nối mạng trong Multiplayer games

Giảng viên hướng dẫn: Phạm Vũ Minh Tú

Nhóm: 10

Sinh viên thực hiện	Mã sinh viên
Bùi Thị Mai	B19DCPT154
Nguyễn Minh Quang	B19DCPT185
Phạm Văn Đang	B19DCPT040
Nguyễn Văn Hưng	B19DCPT115

Hà Nội- 2023

MỤC LỤC

Danh mục hình ảnh

Danh mục các từ viết tắt

STT	Ký hiệu	Chữ viết đầy đủ
1	LAN	Local Area Network
2	WAN	Wide Area Network
3	VPN	Virtual Private Network
4	PAN	Personal Area Network
5	WLAN	Wireless Local Area Network
6	MAN	Metropolitan Area Network
7	TCP/IP	Transmission Control Protocol/Internet Protocol
8	HTTP	Hypertext Transfer Protocol
9	FTP	File Transfer Protocol
10	SMTP	Simple Mail Transfer Protocol
11	DNS	Domain Name System
12	DHCP	Dynamic Host Configuration Protocol
13	SSL	Secure Sockets Layer
14	TLS	Transport Layer Security
15	ICMP	Internet Control Message Protocol
16	ARP	Address Resolution Protocol
17	SSH	Secure Shell Protocol
18	ACK	Acknowledgment Number
19	SYN	Synchronize
20	FIN	Finish
21	PSH	Push
22	URG	Urgent
23	RST	Reset
24	DoS	Denial of Service

25	VPN	Virtual Private Network
26	RPCs	Remote Procedure Calls
27	API	Application Programming Interface
28	DLC	Data Link Control
29	PC	Personal Computer

I. Các khái niệm

1. Network

1.1 Khái niệm

Mạng (network) là một tập hợp các thiết bị và phần mềm được kết nối với nhau để truyền tải thông tin giữa các thiết bị đó. Một mạng bao gồm các thành phần như máy tính, thiết bị định tuyến, công cụ kết nối, dây cáp, đường truyền, giao thức truyền tải dữ liệu và các thành phần khác.

Mục đích của mạng là để truyền tải dữ liệu và chia sẻ tài nguyên giữa các thiết bị trong mạng. Các loại mạng có thể khác nhau về quy mô và phạm vi, từ các mạng máy tính cá nhân nhỏ đến các mạng toàn cầu phức tạp được sử dụng bởi các tổ chức lớn và internet.

Có các loại mạng khác nhau bao gồm mạng máy tính, mạng di động, mạng LAN (Local Area Network), mạng WAN (Wide Area Network), mạng VPN (Virtual Private Network), và internet.

1.2 Phân loại network

Network có thể được phân loại theo quy mô hoặc mục đích sử dụng của chúng. Quy mô của một network được thể hiện bằng khu vực địa lý và số lượng máy tính. Network chủ yếu có 5 loại sau đây:

a. Mạng khu vực cá nhân (PAN - Personal Area Network)

Đây là loại mạng nhỏ nhất và cơ bản nhất, PAN được tạo thành từ một modem không dây, một hoặc nhiều máy tính, điện thoại, máy in, máy tính

bảng, các thiết bị giải trí cá nhân khác v.v. và xoay quanh một cá nhân trong một tòa nhà duy nhất. Các loại mạng này thường được tìm thấy trong các văn phòng hoặc khu dân cư nhỏ và được quản lý bởi một người hoặc tổ chức từ một thiết bị duy nhất.

b. Mạng cục bộ (LAN - Local Area Network)

Mạng LAN là một trong những mạng phổ biến nhất, nguyên bản nhất và đơn giản nhất. Mạng LAN kết nối các nhóm máy tính và thiết bị điện áp thấp với nhau trong khoảng cách ngắn (trong một tòa nhà hoặc giữa một nhóm hai hoặc ba tòa nhà ở gần nhau) để chia sẻ thông tin và tài nguyên. Hai công nghệ quan trọng tham gia vào mạng này là Ethernet và Wi-fi. Ví dụ về mạng LAN là mạng trong gia đình, trường học, thư viện, văn phòng, v.v. Sử dụng bộ định tuyến, mạng LAN có thể kết nối với mạng diện rộng (WAN) để truyền dữ liệu nhanh chóng và an toàn.

c. Mạng diện rộng (WAN - Wide Area Network)

WAN là một loại mạng máy tính kết nối các máy tính trên một khoảng cách địa lý lớn thông qua một đường truyền dùng chung. Nó không bị giới hạn ở một địa điểm duy nhất mà mở rộng ra nhiều địa điểm. WAN cũng có thể được định nghĩa là một nhóm các mạng cục bộ giao tiếp với nhau. Ví dụ phổ biến nhất của WAN là Internet.

d. Mạng cục bộ không dây (WLAN - Wireless Local Area Network)

WLAN là một loại mạng máy tính hoạt động như một mạng cục bộ nhưng sử dụng công nghệ mạng không dây như Wi-Fi. Mạng này không yêu cầu các thiết bị giao tiếp qua cáp vật lý như trong mạng LAN, mà cho phép các thiết bị giao tiếp không dây. Ví dụ phổ biến nhất của WLAN là Wifi.

e. Mạng khu vực đô thị (MAN - Metropolitan Area Network)

MAN lớn hơn mạng LAN nhưng nhỏ hơn mạng WAN. Đây là loại mạng máy tính kết nối các máy tính trên một khoảng cách địa lý thông qua một đường truyền thông tin chung qua thành phố, thị trấn hoặc khu vực đô thị. Ví dụ về MAN là mạng trong các thị trấn, thành phố, một thành phố lớn, một khu vực rộng lớn trong nhiều tòa nhà, v.v.

1.3 Ứng dụng Network vào trong đời sống

Network ngày càng đóng vai trò quan trọng trong bất kỳ tổ chức, doanh nghiệp thậm chí mỗi cá nhân. Dưới đây là một số ứng dụng phổ biến của Network trong đời sống:

- Chia sẻ thông tin và tài nguyên - Network cho phép các doanh nghiệp có các phòng ban ở các vị trí khác nhau chia sẻ thông tin một cách rất hiệu quả. Các máy tính được kết nối với network có thể truy cập vào Các chương trình và phần mềm trong bất kỳ máy tính nào. Network cũng cho phép chia sẻ thiết bị phần cứng, như máy in và máy scan giữa những người dùng khác nhau.
- Truy xuất thông tin từ xa - Thông qua network, người dùng có thể truy xuất thông tin từ xa một cách dễ dàng. Thông tin được lưu trữ trong cơ sở dữ liệu từ xa và người dùng có quyền truy cập thông qua các hệ thống thông tin như World Wide Web.
- Giao tiếp giữa các cá nhân với tốc độ nhanh chóng - Ngày nay network đã tăng tốc độ và khối lượng giao tiếp hơn bao giờ hết. Có thể dễ dàng cho hai hoặc nhiều nhân viên làm việc từ xa trong cùng một dự án bằng cách phân vùng dự án thông qua network.

- Hệ thống có độ tin cậy cao - Bằng cách sử dụng hệ thống network, dữ liệu quan trọng có thể được lưu tại nhiều vị trí. Nếu sự cố xảy ra ở một nguồn, thì hệ thống sẽ vẫn tiếp tục hoạt động và dữ liệu sẽ vẫn có sẵn từ các nguồn khác. Nếu một máy tính bị lỗi hoặc gặp sự cố, dữ liệu có thể được khôi phục từ các máy tính khác của mạng. Bằng cách này, dữ liệu được bảo mật trong mạng.

2. Giao thức kết nối mạng

Giao thức kết nối mạng là một tập hợp các quy tắc và quy định được sử dụng để thiết lập, duy trì và chấm dứt một kết nối mạng giữa hai hoặc nhiều thiết bị trong mạng máy tính. Giao thức kết nối mạng được sử dụng để đảm bảo rằng các thiết bị trong mạng có thể giao tiếp và truyền dữ liệu với nhau một cách hiệu quả và đáng tin cậy.

Các giao thức kết nối mạng thường được phân thành hai loại chính: giao thức phân phối và giao thức truyền tải.

- Giao thức phân phối (routing protocol) được sử dụng để quản lý và phân phối các gói dữ liệu giữa các mạng và các thiết bị định tuyến (router) trong mạng. Giao thức phân phối sẽ xác định đường đi tốt nhất cho các gói dữ liệu để chúng có thể đến được đích một cách nhanh chóng và hiệu quả.
- Giao thức truyền tải (transport protocol) được sử dụng để quản lý việc truyền tải dữ liệu giữa các thiết bị trong mạng. Giao thức truyền tải sẽ đảm bảo rằng dữ liệu được truyền tải đến đích một cách chính xác và đáng tin cậy.

Một số giao thức kết nối mạng phổ biến bao gồm TCP/IP, HTTP, FTP, SMTP, DNS và DHCP. Các giao thức này đóng vai trò quan trọng trong việc kết nối và truyền tải dữ liệu giữa các thiết bị trong mạng.

Trong game multiplayer, các giao thức kết nối mạng được sử dụng để cho phép người chơi trong một trò chơi trực tuyến có thể giao tiếp và tương tác với nhau thông qua mạng Internet. Các giao thức này giúp đảm bảo tính năng lưu lượng mạng, độ trễ thấp và ổn định trong các trò chơi trực tuyến.

3. Lập trình mạng socket

Lập trình với Socket là nền tảng cho việc phát triển tất cả các loại phần mềm có sử dụng dịch vụ truyền thông mạng, cho phép chúng ta sử dụng các dịch vụ truyền thông mạng mà hệ thống cung cấp.

Các ứng dụng mạng đều bao gồm các cặp tiến trình và quá trình truyền thông giữa chúng. Bất kỳ thông điệp nào truyền đi từ 1 tiến trình tới tiến trình còn lại đều phải đi qua mạng. Dữ liệu từ tiến trình tới dịch vụ truyền thông phải đi qua 1 đối tượng trung gian gọi là socket

II. Các loại giao thức kết nối mạng trong multiplayer game

1. TCP/IP

1.1 Khái niệm

TCP/IP (Transmission Control Protocol/Internet Protocol - Giao thức điều khiển truyền nhận/ Giao thức liên mạng) là một bộ giao thức truyền thông được sử dụng để kết nối các thiết bị với internet và truyền dữ liệu qua internet.

TCP/IP bao gồm hai giao thức chính: TCP và IP. TCP cung cấp khả năng phân phối các gói dữ liệu theo thứ tự, đáng tin cậy giữa các thiết bị qua mạng. Nó chia một lượng lớn dữ liệu thành các gói nhỏ hơn, sau đó được truyền và tập hợp lại ở đầu nhận. TCP cũng cung cấp khả năng kiểm soát

luồng và kiểm tra lỗi để đảm bảo dữ liệu được truyền chính xác và không bị thất thoát.

IP cung cấp định tuyến các gói dữ liệu giữa các thiết bị qua internet. Nó xử lý việc đánh địa chỉ của các thiết bị, phân mảnh và lắp ráp lại các gói khi chúng được truyền qua internet.

Trong multiplayer game, TCP/IP được sử dụng để truyền dữ liệu giữa máy khách và máy chủ trò chơi qua internet. Người lập trình sử dụng mã mạng TCP/IP để thiết lập và quản lý kết nối giữa máy khách và máy chủ trò chơi, cũng như để gửi và nhận gói dữ liệu chứa thông tin trạng thái trò chơi, vị trí người chơi, đầu vào của người dùng và các dữ liệu khác liên quan đến trò chơi.

1.2 Mục đích và tính năng của giao thức

Bộ giao thức TCP/IP là một phần cơ bản trong cách vận hành của internet và nhiều mạng máy tính khác. Mục đích của nó là cung cấp một cách thức chuẩn hóa và đáng tin cậy để các thiết bị giao tiếp với nhau qua mạng, bất kể phần cứng hoặc phần mềm được sử dụng bởi những thiết bị.

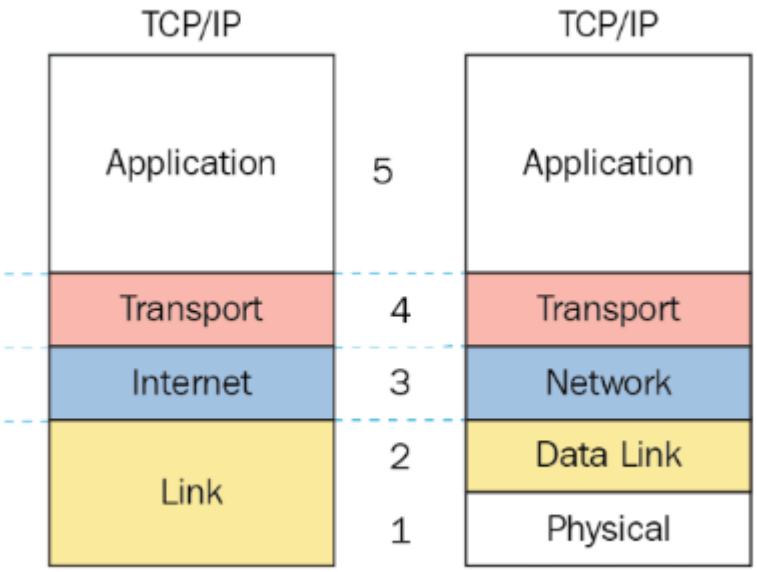
Một số tính năng chính của giao thức TCP/IP bao gồm:

- Định địa chỉ: Giao thức IP cung cấp một cách định địa chỉ các thiết bị trên mạng được tiêu chuẩn hóa, cho phép các gói dữ liệu được định tuyến đến đích chính xác của chúng.
- Độ tin cậy: Giao thức TCP cung cấp khả năng gửi gói dữ liệu đáng tin cậy, đảm bảo rằng chúng được nhận theo đúng thứ tự và không có lỗi.
- Tính linh hoạt: TCP/IP hỗ trợ nhiều loại ứng dụng và thiết bị, làm cho nó trở thành một bộ giao thức đa năng và linh hoạt.

- Khả năng mở rộng: TCP/IP được thiết kế để hoạt động hiệu quả trên các mạng có quy mô khác nhau, từ mạng cục bộ nhỏ đến mạng internet toàn cầu.
- Khả năng tương thích: TCP/IP tương thích với nhiều loại hệ điều hành và phần cứng, cho phép các thiết bị có cấu hình khác nhau giao tiếp với nhau.
- Bảo mật: TCP/IP bao gồm các giao thức như SSL và TLS cung cấp liên lạc an toàn qua internet.

1.3 Cấu trúc của giao thức

Một mô hình TCP/IP tiêu chuẩn bao gồm 4 lớp được chồng lên nhau, bắt đầu từ tầng thấp nhất là Tầng vật lý (Physical) → Tầng mạng (Network) → Tầng giao vận (Transport) và cuối cùng là Tầng ứng dụng (Application).



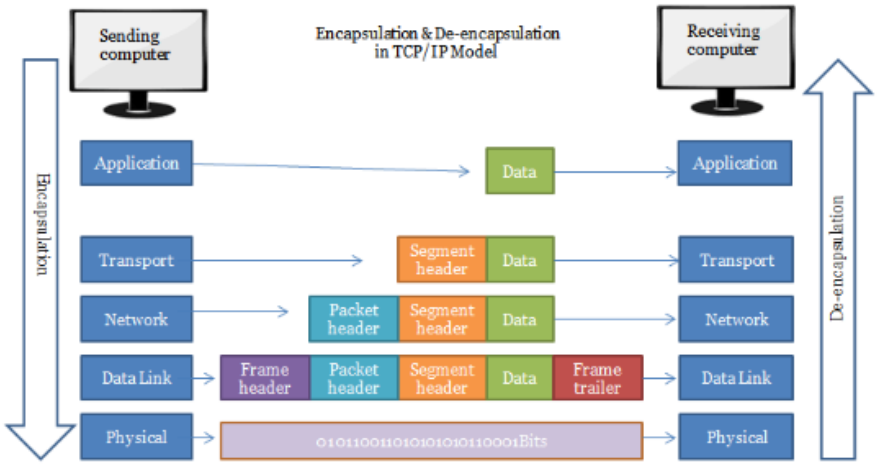
Hình 1. 1: Cấu trúc 1 giao thức TCP/IP

- **Tầng 1 - Tầng Vật lý (Physical)**

Chịu trách nhiệm truyền dữ liệu giữa hai thiết bị trong cùng một mạng. Tại đây, các gói dữ liệu được đóng vào các khung (được gọi là Frame) và được định tuyến đến đích chỉ là ban đầu.

- **Tầng 2 - Tầng mạng (Internet)**

Tại đây các phân đoạn dữ liệu sẽ được đóng gói (Packets) với kích thước mỗi gói phù hợp với mạng chuyển mạch mà nó được sử dụng để truyền dữ liệu. Lúc này, các gói tin được chèn thêm phần Header chứa thông tin của các tầng mạng và tiếp tục được chuyển đến các tầng tiếp theo. Các giao thức chính trong tầng là IP, ICMP và ARP.



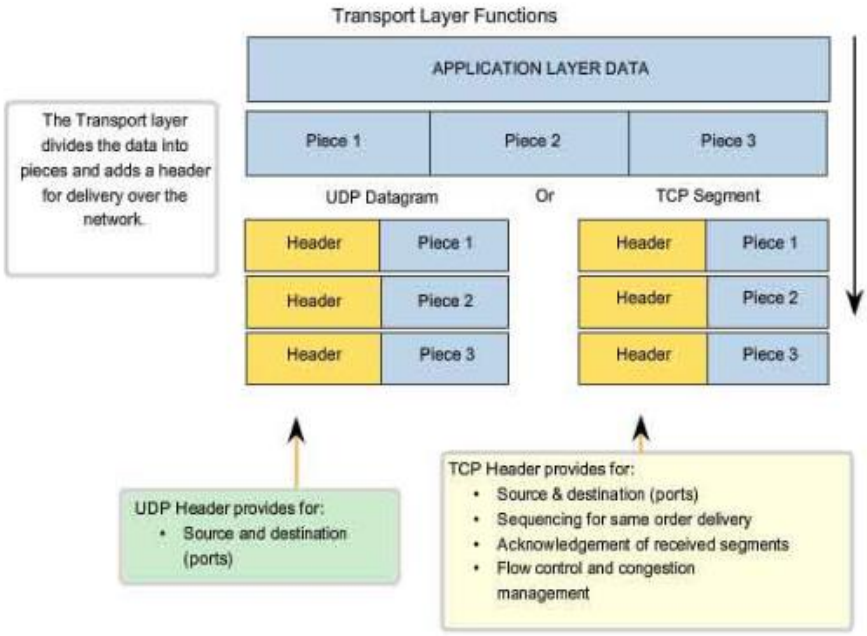
Hình 1. 2: Tầng mạng TCP/IP

- **Tầng 3 - Tầng giao vận (Transport)**

Chức năng chính của tầng 3 là xử lý vấn đề giao tiếp giữa các máy chủ trong cùng một mạng hoặc mạng khác được kết nối với nhau thông qua bộ định tuyến. Tại đây dữ liệu sẽ được phân đoạn, mỗi đoạn sẽ không bằng nhau nhưng kích thước phải nhỏ hơn 64KB. Cấu

hình đầy đủ cấu trúc của một Phân đoạn lúc này là Tiêu đề chứa thông tin điều khiển và sau đó là dữ liệu.

Trong tầng này còn bao gồm 2 giao thức cốt lõi là TCP và UDP. Trong đó, TCP đảm bảo chất lượng gói tin nhưng tiêu tốn thời gian khá lâu để kiểm tra đầy đủ thông tin từ thứ tự dữ liệu cho đến cuộc kiểm tra giám sát vấn đề tắc nghẽn lưu trữ dữ liệu. Trái ngược với điều đó, UDP cho tốc độ truyền tải nhanh hơn nhưng lại không chắc chắn về chất lượng dữ liệu được gửi đi.

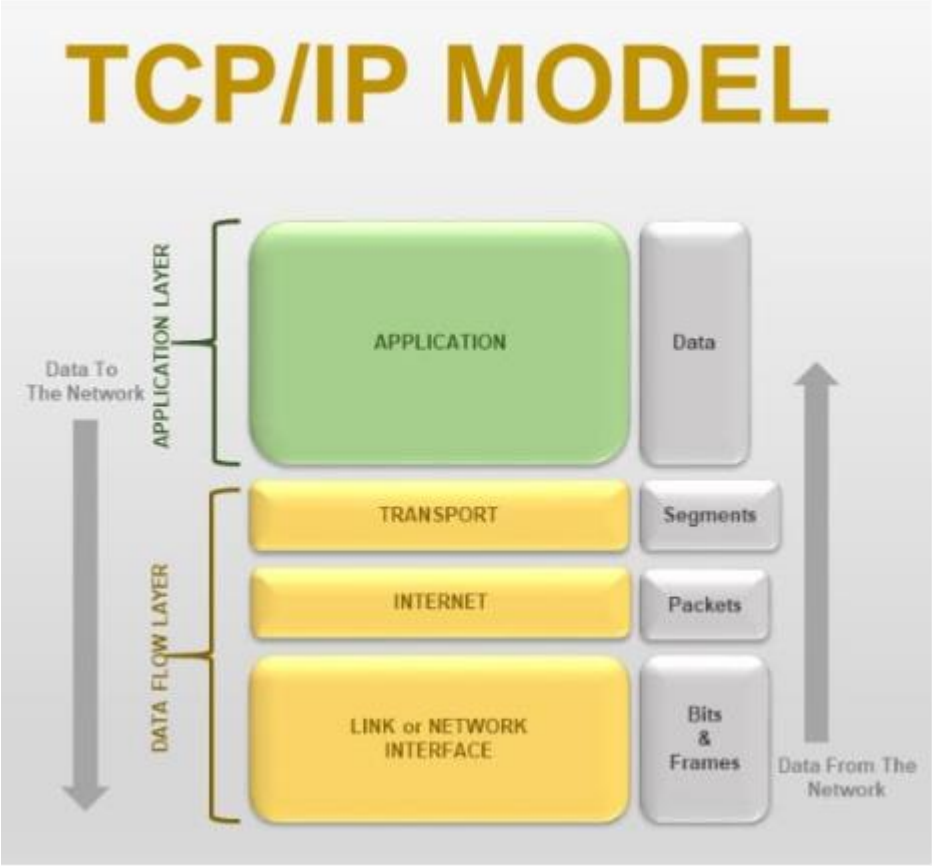


Hình 1. 3: Tầng giao vận TCP/IP

- **Tầng 4 - Tầng Ứng dụng (Application)**

Đây là lớp giao tiếp trên cùng của mô hình. Đúng với tên gọi, Tầng ứng dụng đảm nhận vai trò giao tiếp dữ liệu giữa 2 máy khác nhau thông qua các dịch vụ mạng khác nhau (duyệt web, trò chuyện, gửi email, một số giao thức trao đổi dữ liệu: SMTP, SSH , FPT,...). Dữ liệu khi đến đây sẽ được định dạng theo kiểu Byte nối Byte, cùng

với đó là các thông tin định tuyến giúp xác định đường đi đúng của một gói tin.



Hình 1. 4: Tầng ứng dụng TCP/IP

1.4 Packet

Trong TCP/IP, dữ liệu được truyền dưới dạng các gói tin. Một gói thường bao gồm một tiêu đề và một trọng tải. Tiêu đề chứa thông tin như địa chỉ nguồn và đích, giao thức đang được sử dụng (ví dụ: TCP, UDP) và thông tin điều khiển khác. Tải trọng chứa dữ liệu thực tế được truyền đi.

Khi dữ liệu được gửi từ thiết bị này sang thiết bị khác qua mạng TCP/IP, dữ liệu sẽ được chia thành nhiều gói, với mỗi gói chứa một phần dữ liệu. Các gói này sau đó được gửi qua mạng và được tập hợp lại ở đầu nhận để tạo lại dữ liệu gốc.

Trong TCP, các gói còn được gọi là phân đoạn và chúng được sử dụng để thiết lập kết nối đáng tin cậy giữa hai điểm cuối. Các phân đoạn TCP chứa số thứ tự và xác nhận để đảm bảo phân phối đáng tin cậy và sắp xếp đúng thứ tự dữ liệu.

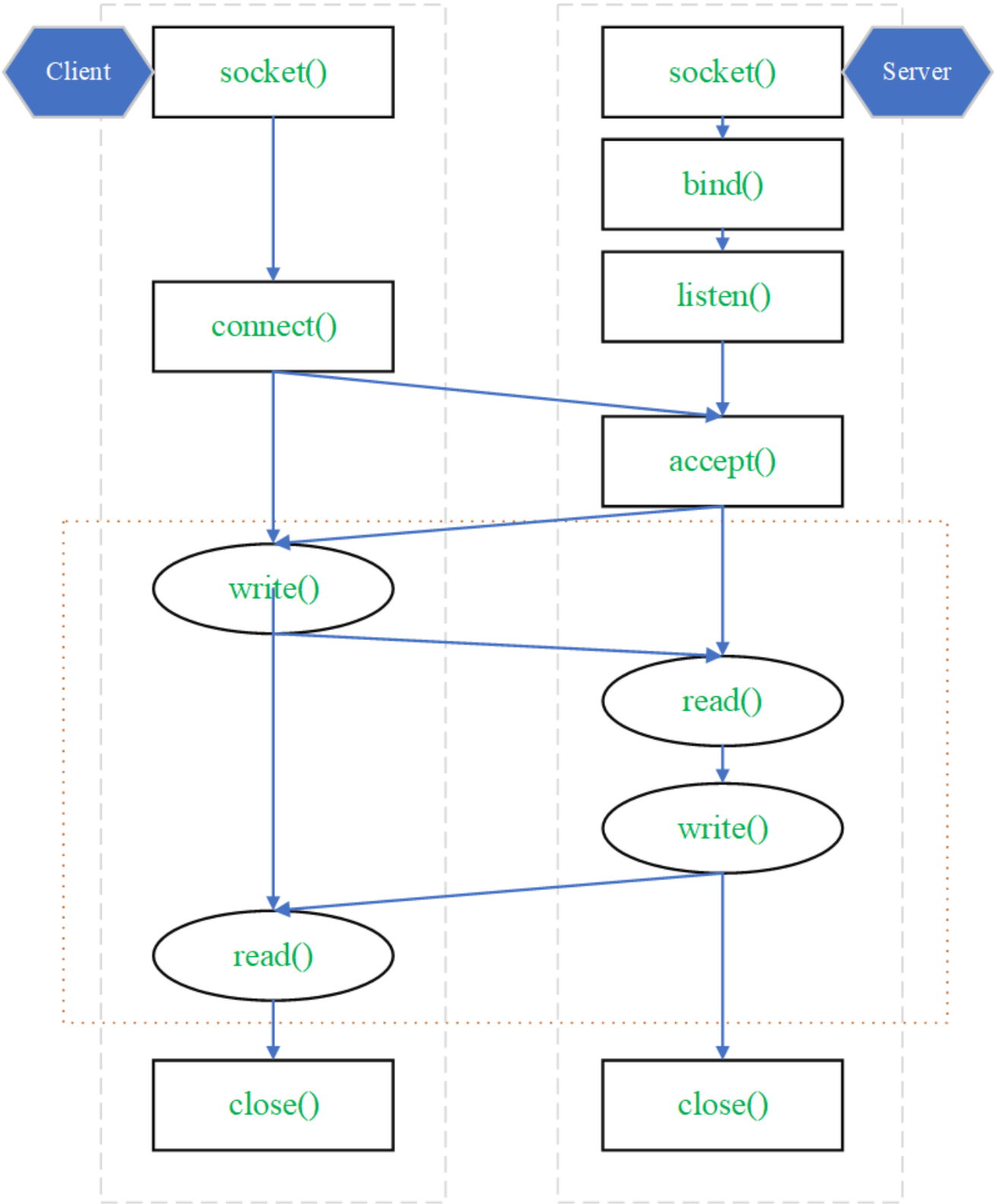
Cấu trúc của gói tin TCP bao gồm các trường sau:

- Port nguồn (Source Port): Đây là số hiệu cổng của ứng dụng nguồn, nó sẽ giúp cho ứng dụng đích nhận biết được gói tin đến từ đâu.
- Port đích (Destination Port): Đây là số hiệu cổng của ứng dụng đích, nó sẽ giúp cho ứng dụng đích biết được gói tin cần đến cho ai.
- Số thứ tự (Sequence Number): Đây là số thứ tự của gói tin, giúp cho các gói tin được sắp xếp theo đúng thứ tự khi truyền tải.
- Số ACK (Acknowledgment Number): Đây là số xác nhận của gói tin, nó cho biết gói tin đã được nhận đến đâu.
- Độ dài tiêu đề (Header Length): Đây là trường chứa độ dài của tiêu đề của gói tin.
- Byte trống (Reserved): Đây là trường dành trống cho mục đích sử dụng trong tương lai.
- Cờ (Flags): Đây là trường chứa các cờ điều khiển, bao gồm SYN, ACK, FIN, PSH, URG, và RST.
- Cửa sổ (Window): Đây là trường chứa kích thước của cửa sổ trượt, cho phép truyền tải dữ liệu theo chu kỳ.
- Kiểm tra định danh (Checksum): Đây là trường chứa giá trị checksum để kiểm tra tính toàn vẹn của gói tin.
- Con trỏ khẩn cấp (Urgent Pointer): Đây là trường chỉ ra vị trí của dữ liệu cần được ưu tiên truyền tải.
- Tùy chọn (Options): Đây là trường tùy chọn cho phép thêm thông tin phụ vào gói tin TCP.

1.5 Tính bảo mật của giao thức

TCP/IP được thiết kế có tính đến bảo mật và bao gồm một số giao thức cung cấp mã hóa, xác thực và tính toàn vẹn dữ liệu cho dữ liệu được truyền qua internet. Tuy nhiên, nó vẫn dễ dàng bị tấn công bởi nhiều kiểu tấn công khác nhau, chẳng hạn như tấn công từ chối dịch vụ (DoS) và đánh hơi gói tin, do đó, điều quan trọng là phải thực hiện các bước bổ sung để bổ sung bảo vệ dữ liệu được truyền qua TCP/IP, chẳng hạn như sử dụng tường lửa và triển khai thực thi mã hóa an toàn.

1.6 Xây dựng giao thức kết nối mạng TCP/IP



Hình 1. 5: Mô hình giao thức TCP/IP

Có thể phân thành 4 giai đoạn như sau:

- **Giai đoạn 1**: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
    - bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
    - listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.
- **Giai đoạn 2**: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.
    - socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
    - connect(): Client gửi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
    - accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.
- **Giai đoạn 3**: Trao đổi thông tin giữa Client và Server.
    - Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gửi đến.
    - Server phân tích và thực thi yêu cầu. Kết quả sẽ được gửi về client bằng lệnh write().
    - Sau khi gửi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().
- **Giai đoạn 4**: Kết thúc phiên làm việc.
    - Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).

- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket g lệnh close().

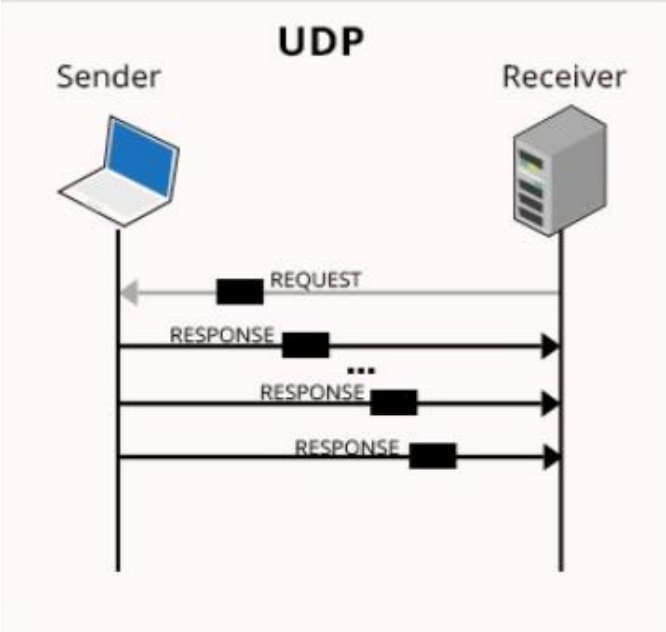
2. UDP

2.1 Khái niệm

User Datagram Protocol (UDP) thường được sử dụng để truyền tải các gói tin dữ liệu giữa các thiết bị trong mạng. UDP được ưa chuộng trong game multiplayer bởi vì nó có tính năng độ trễ thấp và tốc độ truyền tải nhanh, giúp đảm bảo sự liên tục và chính xác của các tác vụ trò chơi.

2.2 Mục đích và tính năng của giao thức

Như đã nói trên, UDP được sử dụng trong game multiplayer vì nó cho phép truyền dữ liệu nhanh chóng và với độ trễ thấp hơn so với TCP. Điều này là quan trọng vì trong các trò chơi đa người chơi, các tín hiệu phải được truyền đi và đến các máy khách càng nhanh càng tốt để tránh độ trễ gây ra sự cố trong trò chơi.



Hình 2. 1: Truyền tải dữ liệu trong UDP

Một số tính năng quan trọng của UDP trong game multiplayer bao gồm:

- Tốc độ truyền dữ liệu nhanh hơn so với TCP.
- Độ trễ thấp hơn, giảm thiểu sự cố liên quan đến độ trễ trong trò chơi.
- Không yêu cầu thiết lập kết nối trước khi truyền dữ liệu, giảm thiểu độ trễ.
- Cho phép truyền dữ liệu nhanh chóng và không đảm bảo độ tin cậy, phù hợp với các trò chơi yêu cầu tốc độ và độ trễ thấp hơn độ tin cậy.
- Sử dụng cổng để xác định ứng dụng nhận dữ liệu, giúp đảm bảo dữ liệu được truyền đến đúng ứng dụng trong máy khách.

Tuy nhiên, do UDP không đảm bảo độ tin cậy và không có cơ chế xử lý lỗi dữ liệu, các lỗi như mất dữ liệu hoặc dữ liệu trùng lặp có thể xảy ra. Để giảm thiểu các sự cố này, các phần mềm game thường sử dụng các kỹ thuật và cơ chế bổ sung để đảm bảo tính đúng đắn của dữ liệu nhận được từ UDP.

2.3 Cấu trúc của giao thức

Cấu trúc của gói tin UDP gồm 4 trường chính

- Port nguồn (Source Port): Đây là số hiệu cổng của ứng dụng nguồn, nó sẽ giúp cho ứng dụng đích nhận biết được gói tin đến từ đâu.
- Port đích (Destination Port): Đây là số hiệu cổng của ứng dụng đích, nó sẽ giúp cho ứng dụng đích biết được gói tin cần đến cho ai.
- Độ dài gói tin (Length): Đây là trường chứa độ dài của toàn bộ gói tin, bao gồm cả tiêu đề và dữ liệu.
- Kiểm tra định danh (Checksum): Đây là trường chứa giá trị checksum để kiểm tra tính toàn vẹn của gói tin.

Hình 2. 2: Cấu trúc giao thức UDP

Các trường này được sắp xếp theo thứ tự từ trái sang phải trong gói tin UDP. Sau các trường này là dữ liệu thực sự được truyền tải trong gói tin.

=> Trong game multiplayer, UDP được sử dụng để truyền tải các thông tin như tọa độ của nhân vật, các hành động của người chơi, v.v. UDP có tính đơn giản và hiệu suất cao hơn TCP, tuy nhiên, nó không đảm bảo tính toàn vẹn dữ liệu và có thể góp phần làm giảm chất lượng trò chơi nếu không được sử dụng đúng cách. Trong trường hợp này, TCP thường được sử dụng thay thế cho UDP

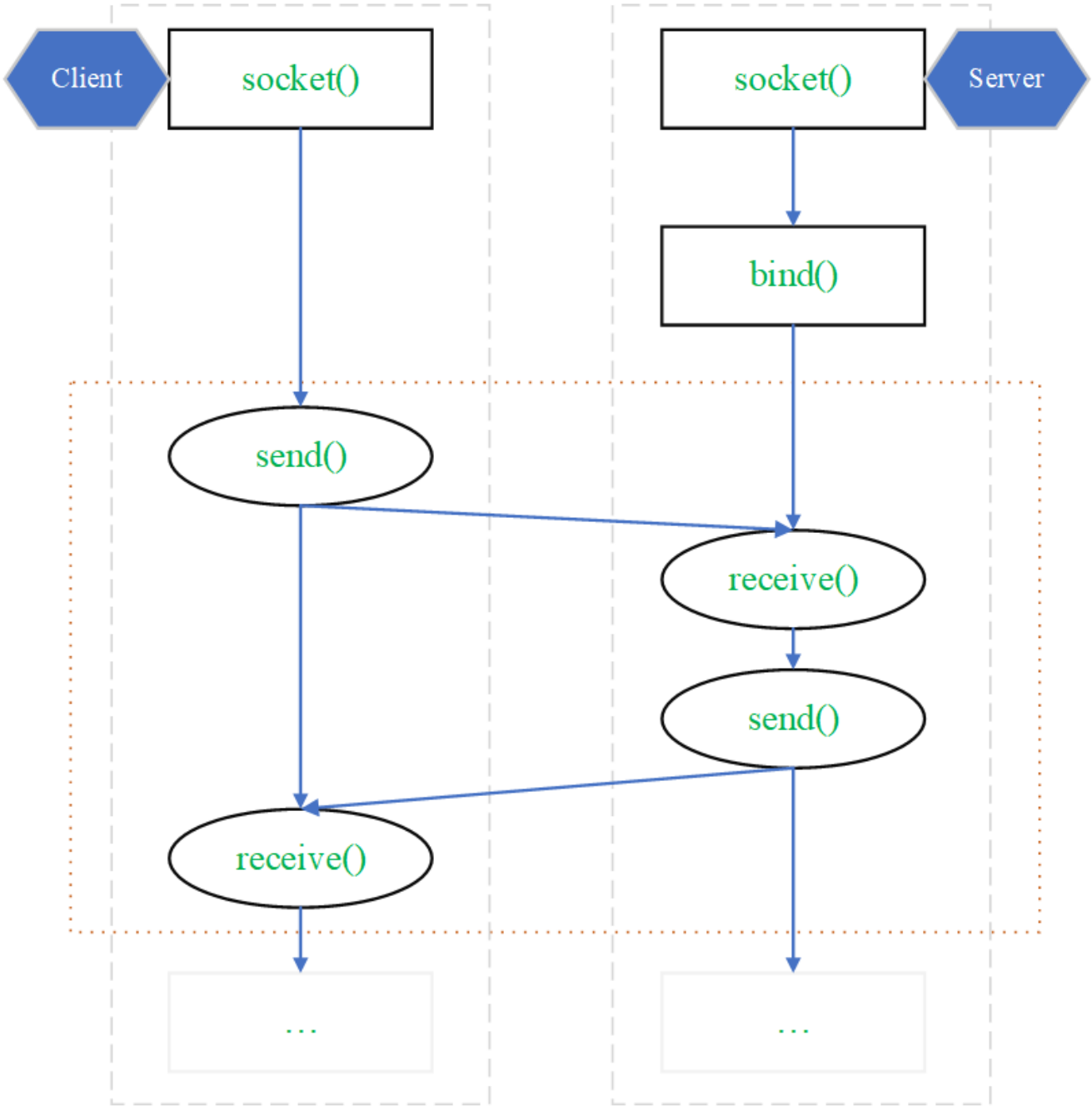
2.4 Packet (gói tin)

Packet của User Datagram Protocol (UDP) trong game multiplayer chứa thông tin để truyền từ máy chủ game đến các máy khách (clients) của người chơi hoặc ngược lại. Mỗi packet UDP bao gồm các trường sau:

- Port nguồn (Source Port): Đây là số cổng được sử dụng bởi máy gửi để gửi packet. Nó giúp máy nhận biết từ đâu packet được gửi đến.
- Port đích (Destination Port): Đây là số cổng được sử dụng bởi máy nhận để nhận packet. Nó cho phép các packet được gửi đến đúng ứng dụng trên máy khách.

- Độ dài (Length): Trường này xác định độ dài của packet tính bằng số byte.
- Checksum: Trường này chứa giá trị checksum (tổng kiểm tra), được sử dụng để kiểm tra tính chính xác của packet. Giá trị checksum được tính toán dựa trên nội dung của packet và được sử dụng để xác định xem package đã bị hỏng hay không.
- Dữ liệu (Data): Trường này chứa dữ liệu cần truyền đi.
- Packet UDP được gửi đi mà không yêu cầu thiết lập kết nối trước khi truyền dữ liệu. Khi packet được nhận, các ứng dụng trên máy khách có thể trích xuất dữ liệu từ packet và sử dụng chúng để hiển thị thông tin hoặc điều khiển các hoạt động trong trò chơi.

2.5 Xây dựng giao thức kết nối mạng UDP/IP



Hình 2. 3: Mô tả gao thức UDP

Có thể phân thành 3 giai đoạn như sau:

- Giai đoạn 1: Server tạo Socket – gán số hiệu cổng.

- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- bind(): Server yêu cầu gán số hiệu cổng cho socket.

- Giai đoạn 2: Client tạo Socket.
  - socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
- Giai đoạn 3: Trao đổi thông tin giữa Client và Server.
  - Sau khi tạo Socket xong, Client và Server có thể trao đổi thông tin qua lại với nhau thông qua hai hàm send() và receive().
  - Đơn vị dữ liệu trao đổi giữa Client và Server là các Datagram Package (Gói tin thư tín).
  - Protocol của ứng dụng phải định nghĩa khuôn dạng và ý nghĩa của các Datagram Package. Mỗi Datagram Package có chứa thông tin về địa chỉ người gửi và người nhận (IP, Port).

2.6 Tính bảo mật

UDP là một giao thức có tính bảo mật kém không đảm bảo việc gửi và nhận các gói tin đúng thứ tự hoặc đảm bảo rằng chúng đã được gửi và nhận một cách an toàn và chính xác.

Do đó, nếu bạn muốn đảm bảo tính bảo mật của ứng dụng sử dụng giao thức UDP, bạn cần sử dụng các giải pháp bảo mật bổ sung như mã hóa và xác thực gói tin để đảm bảo rằng các thông tin được truyền qua UDP không bị xâm nhập hoặc thay đổi bởi người ngoài. Ví dụ, VPN (Virtual Private Network) thường sử dụng UDP để truyền dữ liệu nhưng cung cấp các giải pháp bảo mật bổ sung để đảm bảo an toàn và bảo mật cho dữ liệu truyền qua mạng.

3. WebSockets

3.1 Khái niệm

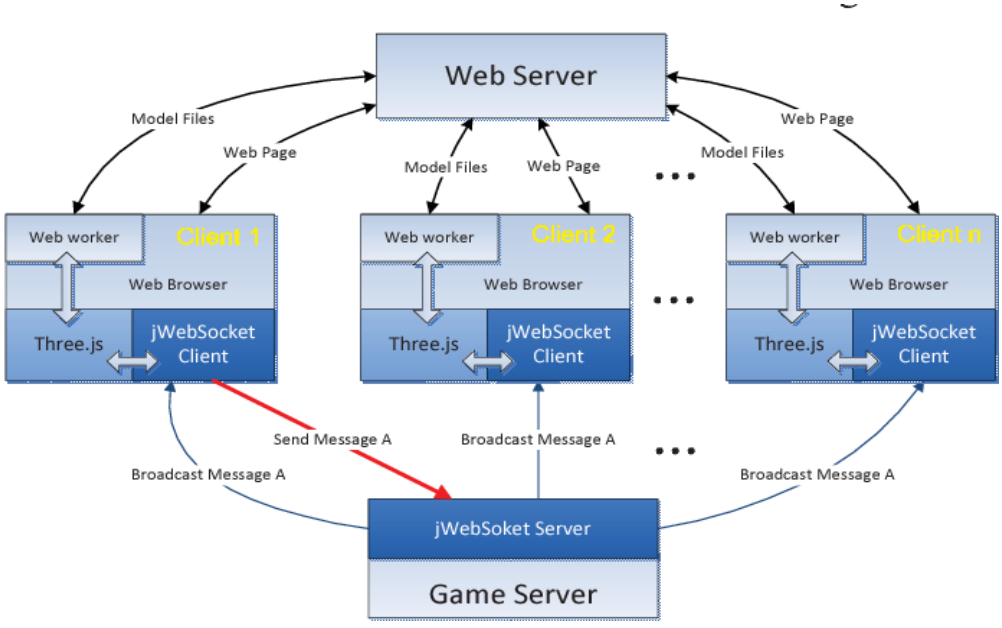
WebSockets là một công nghệ được sử dụng để tạo kết nối hai chiều giữa trình duyệt và máy chủ. Nó cho phép truyền tải dữ liệu thời gian thực trong thời gian thực qua một kết nối duy nhất, đồng thời giảm thiểu độ trễ và số lượng lưu lượng mạng cần thiết để truyền tải dữ liệu.

3.2 Mục đích và tính năng của giao thức

Trong game multiplayer, WebSockets thường được sử dụng để truyền tải dữ liệu game giữa trình duyệt và máy chủ một cách nhanh chóng và hiệu quả. Việc sử dụng WebSockets cho phép trò chơi đáp ứng nhanh hơn với các tương tác của người chơi và đảm bảo tính thời gian thực của trò chơi.

3.3 Cấu trúc của giao thức

Cấu trúc của WebSockets trong game multiplayer bao gồm ba thành phần chính: máy chủ WebSocket, trình duyệt, và giao thức WebSocket.



Hình 3. 1: Cấu trúc giao thức WebSockets

Máy chủ WebSocket: Đây là phần của hệ thống game multiplayer nhận và xử lý các yêu cầu của trình duyệt thông qua giao thức WebSocket. Máy chủ này sẽ tạo và duy trì các kết nối WebSocket với trình duyệt của người chơi và xử lý các tương tác của họ trong trò chơi.

Trình duyệt: Trình duyệt sử dụng giao thức WebSocket để thiết lập kết nối với máy chủ WebSocket và truyền tải dữ liệu game qua kết nối đó. Người chơi có thể tương tác với trò chơi thông qua giao diện người dùng của trình duyệt và các tương tác này sẽ được gửi đến máy chủ thông qua kết nối WebSocket.

Giao thức WebSocket: Đây là một giao thức mở cho phép truyền tải dữ liệu theo hai chiều qua một kết nối TCP. Giao thức WebSocket cho phép truyền tải dữ liệu thời gian thực trong trò chơi multiplayer và giảm thiểu độ trễ.

Khi trò chơi được bắt đầu, trình duyệt của người chơi sẽ thiết lập kết nối WebSocket với máy chủ. Sau khi kết nối đã được thiết lập, trình duyệt có thể gửi dữ liệu đến máy chủ thông qua kết nối WebSocket. Máy chủ sẽ xử lý và đáp ứng các yêu cầu từ trình duyệt thông qua kết nối WebSocket.

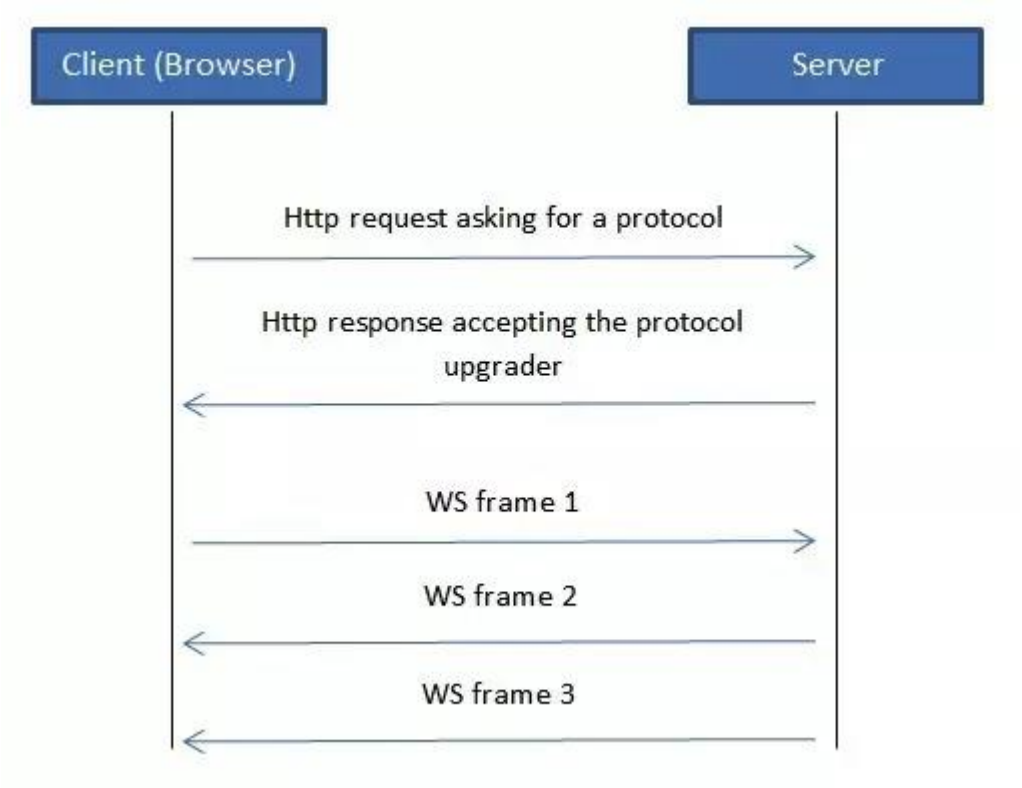
Tóm lại, cấu trúc của WebSockets trong game multiplayer bao gồm máy chủ WebSocket, trình duyệt và giao thức WebSocket. Kết nối WebSocket cho phép truyền tải dữ liệu thời gian thực trong trò chơi và giảm thiểu độ trễ.

3.4 Packet

Packet của WebSocket trong game multiplayer bao gồm hai thành phần chính: header và payload.

- **Header**: Header của một packet WebSocket bao gồm các thông tin về loại dữ liệu truyền tải, độ dài của dữ liệu, mã hóa, xác thực, ... Header còn chứa các trường khác như địa chỉ nguồn và đích, phân đoạn, cờ, số thứ tự và checksum.
- **Payload:** Payload của một packet WebSocket chứa dữ liệu được truyền tải giữa trình duyệt và máy chủ. Payload có thể là bất kỳ loại dữ liệu nào cần truyền tải trong trò chơi multiplayer, chẳng hạn như các tương tác của người chơi, trạng thái của trò chơi, vị trí của các đối tượng trong trò chơi, và nhiều hơn nữa.

3.5 Cách hoạt động của websocket



Hình 3. 2: Mô tả quá trình truyền tải dữ liệu của WebSockets

Giao thức có hai phần: Bắt tay và truyền dữ liệu Ban đầu client sẽ gửi yêu cầu khởi tạo kết nối websocket đến server, server kiểm tra và gửi trả kết quả chấp

nhận kết nối, sau đó kết nối được tạo và quá trình gửi dữ liệu có thể được thực hiện, dữ liệu chính là các Ws frame

- Kết nối:

- Đầu tiên client sẽ gửi một http request yêu cầu nâng cấp

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Hình 3. 3: Request client gửi về server trong WebSockets

- Server trả về:

```
HTTP/1.1 101 Switching Protocols
 Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Hình 3. 4: Server trả về request trong WebSockets

- Để xác nhận việc kết nối, client sẽ gửi một giá trị Sec-WebSocket-Key được mã hóa bằng Base64 đến server.
- Sau đó bên server sẽ thực hiện:
  + Nối thêm chuỗi cố định là “258EAFA5-E914-47DA-95CA-C5AB0DC85B11″ vào Sec-WebSocket-Key để được chuỗi mới là “x3JJHMbDL1EzLkh9GBhXDw==258EAFA5-E914-47DA-95CA-C5AB0DC85B11″.
  + Thực hiện mã hóa SHA-1 chuỗi trên để được “1d29ab734b0c9585240069a6e4e3e91b61da1969″.

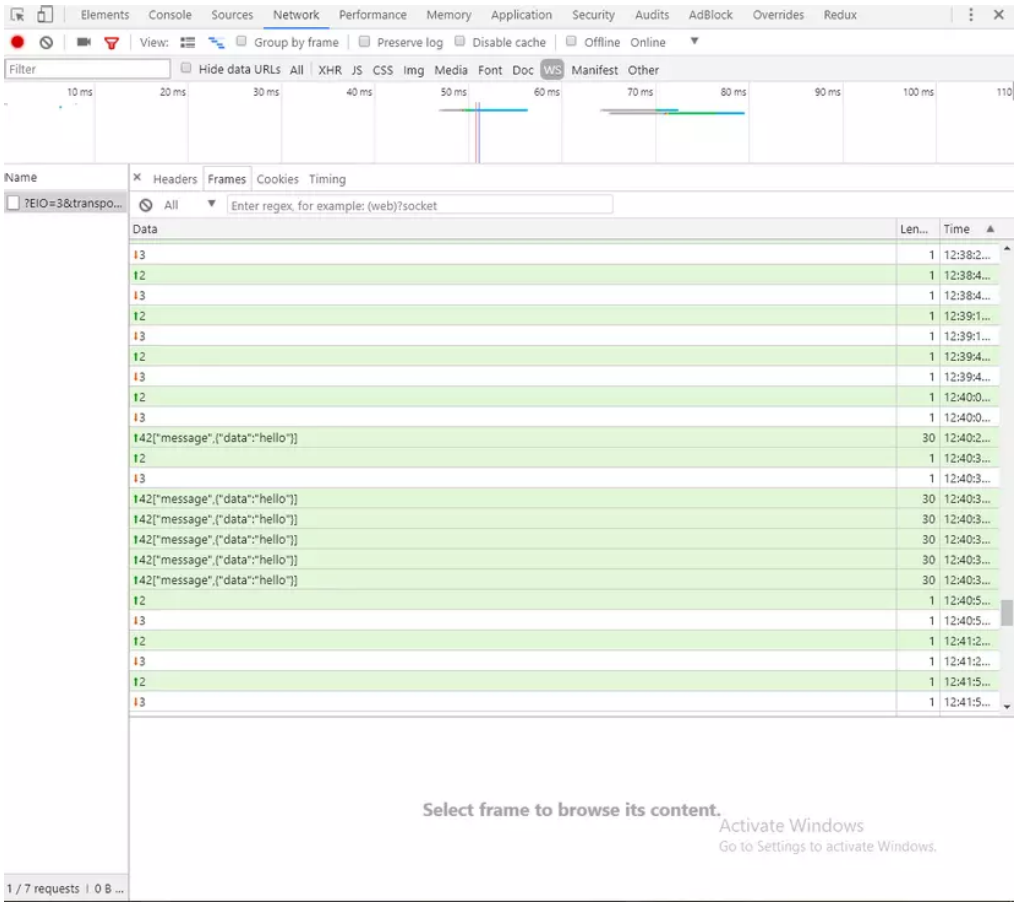
+ Mã hóa kết quả vừa nhận được bằng Base64 để được "HSmrc0sMlYUkAGmm5OPpG2HaGWk="
+ Gửi response lại client kèm với giá trị Sec-WebSocket-Accept chính là chuỗi kết quả vừa tạo ra.

- Client sẽ kiểm tra status code (phải bằng 101) và Sec-WebSocket-Accept xem có đúng với kết quả mong đợi không và thực hiện kết nối.

Hình 3. 5: Kết quả kết nối trong WebSockets

- Truyền dữ liệu

  Dữ liệu sẽ được truyền thông qua một kết nối duy nhất được tạo ra sau quá trình bắt tay. Dữ liệu được truyền bằng các Frame, ta có thể thấy nó khi bật trình debug của trình duyệt lên



Hình 3. 6: Truyền dữ liệu trong WebSockets

3.6 Tính bảo mật của giao thức

WebSocket được sử dụng trong nhiều ứng dụng web, bao gồm cả trò chơi đa người chơi (multiplayer games). Tuy nhiên, việc đảm bảo tính bảo mật của WebSocket trong game multiplayer cũng là một vấn đề quan trọng cần được xem xét.

WebSocket có thể được sử dụng để truyền tải dữ liệu nhạy cảm, chẳng hạn như thông tin về tài khoản người dùng và các thông tin cá nhân khác.

4. HTTP

4.1 Khái niệm

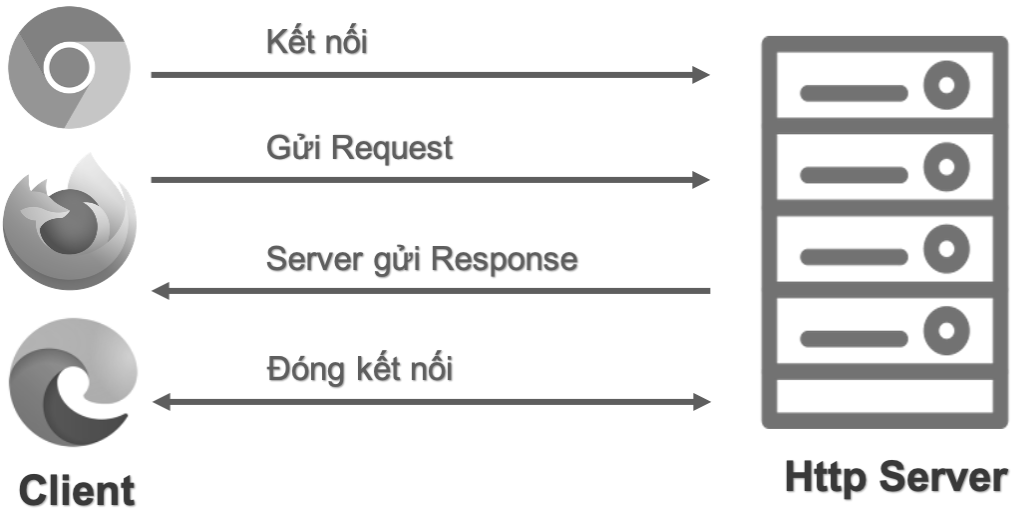
HTTP (Hypertext Transfer Protocol) là một giao thức truyền tải siêu văn bản được sử dụng để truyền tải dữ liệu qua mạng Internet. Tuy nhiên, HTTP không phải là một giao thức thích hợp cho game multiplayer vì nó thiếu tính năng thời gian thực và độ trễ thấp.

Trong game multiplayer, thời gian đáp ứng (latency) và độ trễ (lag) là hai yếu tố quan trọng để đảm bảo trò chơi được diễn ra một cách mượt mà và thú vị. Việc sử dụng HTTP có thể gây ra độ trễ và thời gian đáp ứng chậm do các yếu tố như đóng gói dữ liệu và xác thực.

Thay vì sử dụng HTTP, game multiplayer thường sử dụng các giao thức đặc biệt như User Datagram Protocol (UDP) hoặc Transmission Control Protocol (TCP) để đảm bảo tính thời gian thực và độ trễ thấp. UDP thường được sử dụng trong các trò chơi đa người chơi để truyền tải dữ liệu với tốc độ cao và độ trễ thấp, trong khi TCP được sử dụng trong các trò chơi yêu cầu tính toàn vẹn dữ liệu như trò chơi trực tuyến bài.

Tóm lại, HTTP không phải là một giao thức thích hợp để sử dụng trong game multiplayer do thiếu tính năng thời gian thực và độ trễ thấp. Các giao thức khác như UDP hoặc TCP thường được sử dụng thay thế để đảm bảo tính mượt mà và thú vị của trò chơi.

4.2 Truyền tải dữ liệu client- server trong giao thức HTTP



Hình 4. 1: Truyền dữ liệu trong giao thức HTTP

- **Bước 1:** Mở kết nối TCP - Kết nối TCP (Giao thức HTTP dựa trên TPC) trên địa chỉ xác định bởi URL (Uniform Resource Locator) và cổng thường là 80, kết nối này được dùng để gửi các yêu cầu (request) và nhận phản hồi (response). Client có thể mở ra kết nối TCP mới hoặc sử dụng kết nối đang có, thậm chí nó tạo ra nhiều kết nối TCP cùng lúc đến server.
- **Bước 2:** Gửi HTTP Message (request) - HTTP Message (request) chính là nội dung yêu cầu được client tạo ra và gửi đến server. HTTP Message có nội dung text (plain text) mà chúng ta có thể đọc được (người đọc được). Với phiên bản HTTP/2 thì nội dung HTTP Message được bao bọc trong các frame, nó làm cho người không đọc được một cách trực tiếp - tuy nhiên về mặt ý nghĩa nội dung không đổi so với HTTP/1.1
- **Bước 3:** Đọc HTTP Message nhận được từ server (response) - Http Message (response) trả về từ server có cấu trúc tương tự Http Message (request

- **Bước 4:** Đóng kết nối hoặc sử dụng lại cho các truy vấn khác

III. Những đơn vị cung cấp giao thức mạng

Có nhiều đơn vị cung cấp giao thức mạng trong game multiplayer. Dưới đây là một số đơn vị phổ biến:

1. Unity Networking

Unity Networking là một phần của Unity Engine cho phép game developer tạo ra các trò chơi multiplayer trên nền tảng desktop, mobile, và console. Unity Networking cung cấp cho developer các công cụ để tạo ra các kết nối giữa các máy tính hoặc thiết bị di động, trao đổi dữ liệu, và đồng bộ hóa trò chơi giữa các máy tính trong một mạng lưới.

Các thành phần chính của Unity Networking bao gồm:

- Network Manager: Là một component có thể được gắn vào GameObject trong trò chơi của bạn, cho phép bạn quản lý các kết nối mạng và phân phối dữ liệu giữa các client và server.
- Networked Objects: Là các đối tượng trong trò chơi của bạn được đồng bộ hóa giữa các client và server. Các networked objects có thể được đồng bộ hóa để đảm bảo rằng tất cả các client đang chơi trò chơi của bạn thấy cùng một trạng thái và hành động.
- Remote Procedure Calls (RPCs): Là một cơ chế để gọi các phương thức từ client đến server hoặc ngược lại. RPCs có thể được sử dụng để gửi các yêu cầu giữa các client và server để cập nhật trạng thái trò chơi, xử lý hành động của người chơi, v.v.
- Network Identity: Là một thành phần cho phép bạn gắn các ID định danh duy nhất cho các đối tượng trong trò chơi của bạn, cho phép các client biết đối tượng nào đang được đồng bộ hóa.

- Network Transform: Là một thành phần cho phép bạn đồng bộ hóa chuyển động và vị trí của các đối tượng giữa các client và server.

2. Photon

Trong lập trình game multiplayer, Photon là một platform được sử dụng để phát triển các game online trực tuyến. Platform này cung cấp cho nhà phát triển các công cụ để kết nối, đồng bộ hóa và quản lý các kết nối mạng trong game, cho phép nhiều người chơi truy cập và tương tác với nhau trên cùng một trò chơi.

Photon hỗ trợ các giao thức mạng như TCP, UDP, WebSockets và cung cấp các tính năng như phân tích dữ liệu, đồng bộ hóa, chống giật, cân bằng tải và độ trễ. Nó cũng có các công cụ quản lý như bảng điều khiển quản lý, giao diện lập trình ứng dụng (API) dễ sử dụng và một thư viện Unity để tích hợp vào trò chơi.

Các ứng dụng của Photon trong lập trình game bao gồm:

- Điều khiển đồng bộ hóa: Photon giúp đồng bộ hóa trạng thái của game trên nhiều thiết bị và đảm bảo rằng các tín hiệu đến từ người chơi được xử lý một cách hợp lý. Ví dụ, khi một người chơi thực hiện một hành động, Photon đảm bảo rằng hành động đó được đồng bộ hóa và phản ánh trên tất cả các thiết bị khác.
- Kết nối mạng: Photon cung cấp một giao diện lập trình ứng dụng (API) để xử lý các kết nối mạng trong game, cho phép các người chơi kết nối với nhau thông qua các máy chủ hoặc peer-to-peer.
- Quản lý game: Photon cung cấp các công cụ để quản lý các trò chơi trực tuyến, cho phép nhà phát triển quản lý thông tin về người chơi, phòng chơi và các trạng thái khác của game.

Photon được sử dụng rộng rãi trong các trò chơi trực tuyến như Fortnite, Among Us và Minecraft để hỗ trợ việc kết nối và đồng bộ hóa giữa các người chơi trên toàn cầu.

3. Unreal Engine

Unreal Engine là một hệ thống game engine được phát triển bởi Epic Games, và nó cũng cung cấp một số giao thức mạng để phát triển các trò chơi trực tuyến. Các giao thức mạng được hỗ trợ bao gồm:

- Unreal Networking: Đây là một giao thức mạng được xây dựng sẵn trong Unreal Engine và được thiết kế để hỗ trợ các trò chơi đa người chơi trực tuyến. Unreal Networking hỗ trợ các tính năng như cập nhật trạng thái game và đồng bộ hóa các hành động giữa các người chơi.
- Replication: Unreal Engine hỗ trợ replication cho các đối tượng trong game, giúp đồng bộ hóa các trạng thái giữa server và client. Các đối tượng được đồng bộ hóa này có thể bao gồm nhân vật, vật phẩm, vũ khí, v.v.

- RPC (Remote Procedure Call): RPC là một công nghệ mạng cho phép client gửi yêu cầu đến server và server phản hồi lại với kết quả. Unreal Engine hỗ trợ việc triển khai các hàm RPC trên các đối tượng trong game, cho phép client gọi các hàm này trên server.
- Steamworks: Unreal Engine cũng hỗ trợ Steamworks, platform của Valve Corporation để phát triển và quản lý các trò chơi trực tuyến trên Steam. Steamworks cung cấp cho các nhà phát triển một số công cụ và dịch vụ để giúp họ tạo ra các trò chơi trực tuyến.

- Socket.IO: Socket.IO là một thư viện JavaScript cho phép việc truyền thông tin giữa server và client thông qua giao thức WebSockets hoặc Polling. Unreal Engine hỗ trợ việc tích hợp Socket.IO vào các trò chơi trực tuyến.

Các giao thức mạng trong Unreal Engine được sử dụng để tạo ra các trò chơi trực tuyến với các tính năng đa người chơi như chơi đối kháng, chế độ chơi độc đối, chia sẻ thông tin và hỗ trợ cho việc tạo ra các mod và phiên bản tùy chỉnh của trò chơi.

4. Mirror

Mirror là một thư viện mạng miễn phí cho Unity, cho phép các nhà phát triển tạo game multiplayer với các tính năng như đồng bộ hóa và giao tiếp mạng. Nó hỗ trợ cả TCP và UDP.

Về cơ bản, mirror hoạt động bằng cách sao chép trạng thái của trò chơi từ máy chủ (server) đến các máy khách (client) thông qua một giao thức mạng đồng bộ hóa. Khi một sự kiện xảy ra trong trò chơi, như người chơi di chuyển hoặc tấn công, thông tin sẽ được gửi đến máy chủ và được xử lý tại đó. Sau đó, máy chủ sẽ gửi trạng thái mới nhất đến các máy khách để cập nhật trên màn hình của họ.

Trong thư viện Mirror, chỉ có một máy tính (máy chủ) tính toán và quản lý trạng thái của trò chơi, các máy tính khác (máy khách) sẽ nhận thông tin về trạng thái mới nhất từ máy chủ. Điều này giúp giảm độ trễ và đảm bảo rằng các máy tính trong mạng đồng bộ với nhau để tạo ra một trò chơi đồng nhất.

Tuy nhiên, Mirror cũng có nhược điểm khi tạo ra một mức độ phụ thuộc lớn vào máy chủ. Nếu máy chủ bị quá tải hoặc có vấn đề về kết nối, trò chơi có thể trở nên chậm hoặc không ổn định. Do đó, các lập trình viên game cần phải cân nhắc kỹ về cách triển khai mirror để đảm bảo hiệu quả và tính ổn định của trò chơi.

5. Steamworks

Steamworks là một platform cung cấp bởi Valve Corporation để phát triển và quản lý các trò chơi trực tuyến trên Steam, một nền tảng phân phối trò chơi kỹ thuật số. Steamworks cung cấp cho các nhà phát triển một số công cụ và dịch vụ để giúp họ tạo ra các trò chơi trực tuyến, bao gồm:

- Quản lý người chơi và tài khoản: Steamworks cung cấp một hệ thống quản lý người chơi và tài khoản, cho phép nhà phát triển tạo và quản lý các tài khoản người chơi trên Steam.
- Hệ thống chơi đối kháng: Steamworks cung cấp một hệ thống chơi đối kháng cho các trò chơi trực tuyến, bao gồm khả năng tìm kiếm trận đấu, kết nối người chơi và hỗ trợ cho chế độ chơi độc đối.

- Hệ thống cộng đồng: Steamworks cung cấp các công cụ và dịch vụ để tạo ra một cộng đồng trò chơi trực tuyến, bao gồm hệ thống lời nhắn, cửa hàng trong game, hệ thống chia sẻ thông tin và hỗ trợ cho việc tạo ra các mod và phiên bản tùy chỉnh của trò chơi.
- Quản lý DLC và nội dung tải về: Steamworks cung cấp các công cụ và dịch vụ để quản lý các bản cập nhật, DLC và nội dung tải về cho các trò chơi trực tuyến, bao gồm việc phân phối, quản lý và theo dõi các tài nguyên và phiên bản khác nhau của trò chơi.

Steamworks sử dụng một số giao thức mạng như TCP/IP, UDP và WebSockets để kết nối các người chơi với nhau và với các máy chủ trò chơi. Steamworks cũng hỗ trợ việc tạo ra các mạng LAN ảo cho các trò chơi trực tuyến, giúp các người chơi kết nối với nhau trong cùng một mạng LAN dù không cùng một địa chỉ vật lý

6. SmartfoxServer

SmartFoxServer là một platform cung cấp giao thức mạng cho các trò chơi đa người chơi. Nó cung cấp các giao thức mạng như TCP và UDP để cho các trò chơi đa người chơi có thể kết nối và truyền tải dữ liệu một cách ổn định và đáng tin cậy.

Với SmartFoxServer, các lập trình viên có thể dễ dàng thiết lập các kết nối mạng giữa các người chơi và phát triển các tính năng cho trò chơi như chơi đồng thời, chat, đồng bộ hóa dữ liệu và phân tích dữ liệu. SmartFoxServer hỗ trợ các giao thức mạng như socket, HTTP và WebSocket, cung cấp khả năng kết nối và truyền tải dữ liệu linh hoạt cho các trò chơi đa người chơi trên nhiều nền tảng, bao gồm PC, di động và web.

Ngoài ra, SmartFoxServer cũng cung cấp các tính năng quản lý dữ liệu để giúp đảm bảo tính đồng bộ và ổn định cho các trò chơi đa người chơi. Các tính năng này bao gồm đồng bộ hóa dữ liệu, quản lý tài nguyên, chống giật và bảo mật.

SmartFoxServer có khả năng mở rộng và hiệu suất cao, có thể xử lý hàng ngàn người dùng cùng lúc và hỗ trợ cơ chế phân tán để mở rộng hệ thống.

Tóm lại, SmartFoxServer là một đơn vị cung cấp giao thức mạng ổn định và hiệu quả cho các trò chơi đa người chơi, giúp cho các trò chơi đa người chơi có thể kết nối và truyền tải dữ liệu một cách ổn định và đáng tin cậy trên nhiều nền tảng khác nhau.