Multiplayer Games

Joseph Claver

September 23, 2022
Outline:

- Brief History of Multiplayer
- Multiplayer Technology
- Dealing with Latency
- Addendum: Reducing Network Traffic
Brief History of Multiplayer
Early multiplayer games were offline and local

- **Tennis for Two** (1958)
- **Spacewar!** (1962)
- **Pong** (1972)

**Figure 1:** Tennis for Two, played on an oscilloscope!

**Figure 2:** Pong
Where did it start?

Early multiplayer games were offline and local

- Tennis for Two (1958)
- Spacewar! (1962)
- Pong (1972)

Figure 1: Tennis for Two, played on an oscilloscope!

Figure 2: Pong
Where did it start?

Early multiplayer games were offline and local

- Tennis for Two (1958)
- Spacewar! (1962)
- Pong (1972)

Figure 1: Tennis for Two, played on an oscilloscope!

Figure 2: Pong
Soon, wired computer networks (and games) came about!

- PLATO Empire (1973)
- MUD (1978), and MAD (1984)

**Figure 3:** PLATO Empire
Soon, wired computer networks (and games) came about!

- PLATO Empire (1973)
- MUD (1978), and MAD (1984)
Previously, people just connected multiple terminals to a host computer. What if we connected two host computers?

- Maze War (1973)
- SGI Dogfight (1983)

**Figure 4:** SGI Dogfight
Connecting Multiple Computers

Previously, people just connected multiple terminals to a host computer. What if we connected two host computers?

- Maze War (1973)
- SGI Dogfight (1983)

Figure 4: SGI Dogfight
Moving into the modern day, many games use a client-server model, more resembling the original terminal-mainframe model.

- The client sends their inputs over the Internet to a server
  - The server processes the input
  - The server sends all users the result
Moving into the modern day, many games use a client-server model, more resembling the original terminal-mainframe model.

- The client sends their inputs over the Internet to a server
- The server processes the input
- The server sends all users the result
Moving into the modern day, many games use a client-server model, more resembling the original terminal-mainframe model.

- The client sends their inputs over the Internet to a server
- The server processes the input
- The server sends all users the result

**Figure 5:** A Simple Client-Server Diagram
Multiplayer Technology
When transmitting network traffic over the Internet, there are 2 main protocols to use:

TCP (Transmission Control Protocol)
- Connection-oriented
- Guarantees packet order
- Re-transmits lost packets
- Slower

UDP (User Datagram Protocol)
- Not connection-oriented
- Doesn’t guarantee packet order
- Doesn’t re-transmit lost packets
- Faster
Which one should you use?

It depends!

- UDP is good for real-time events
- TCP is good for discrete, stateful events

Which do you think is better for video games?

Figure 6: UDP and TCP
Which one should you use?

It depends!

- UDP is good for real-time events
- TCP is good for discrete, stateful events

Which do you think is better for video games?

**Figure 6:** UDP and TCP
In practice, both UDP and TCP are used!

- UDP is good for real-time state (player position, inputs, etc)
- TCP is good for discrete state changes (score, lives, etc)

It all depends on required speed and reliability.

**Figure 7:** Places to use UDP and TCP
In fact, many game engines will build protocols from both UDP and TCP!

Godot Docs: "Because of the large difference in performance, it often makes sense to re-build the parts of TCP wanted for games [in UDP], while avoiding the unwanted parts"

Case Study: Minecraft

So, in a game like Minecraft, what might we use for the following features?

- Player movement
- Block placing/breaking
- Chatting with other players
- Fighting enemies
The answers:

- Player movement: TCP
- Block placing/breaking: TCP
- Chatting with other players: TCP
- Fighting enemies: TCP

Literally everything is over TCP (at least in Java Edition). Probably not optimal, but fun to think about!

https://wiki.vg/Protocol
A more interesting example in practice is Team Fortress 2.

- Player movement: UDP
- Equipment usage: UDP
- Chatting: UDP
- Remote Console (RCON): TCP

Figure 8: Team Fortress 2 Gameplay
A more interesting example in practice is Team Fortress 2.

- Player movement: UDP
- Equipment usage: UDP
- Chatting: UDP
- Remote Console (RCON): TCP

*Figure 8: Team Fortress 2 Gameplay*
A Better Case Study: Team Fortress 2

A more interesting example in practice is Team Fortress 2.

- Player movement: UDP
- Equipment usage: UDP
- Chatting: UDP
- Remote Console (RCON): TCP

Figure 8: Team Fortress 2 Gameplay
A more interesting example in practice is Team Fortress 2.

- Player movement: UDP
- Equipment usage: UDP
- Chatting: UDP
- Remote Console (RCON): TCP

Figure 8: Team Fortress 2 Gameplay
Dealing with Latency
Whether we use UDP or TCP, there will always be some network delay if we’re working on the Internet.

- Average ping between London and Denver: 119ms
- Average human reaction time: 273ms

(The average ping is from a networking company - likely very close to a best-case scenario!)

https://wondernetwork.com/pings/London

https://humanbenchmark.com/tests/reactiontime/
Dealing with latency

Either way, network delay, or lag, can noticeably affect gameplay. What would you do to "deal with" latency in an online video game?

Figure 9: Victim of Severe Lag (2016, colorized)
Approach 1: Regional Servers

One way to deal with latency is to try to reduce latency!

Many games will group players into regional servers, so that data never has to travel too far, thereby reducing latency.

Figure 10: Regional Servers for Valorant
Approach 2: Client-Side Prediction

In our client-server model, the server owns all world-state. But, we want the user to jump as they hit the button! An idea:

- Simulate the game on client-side as well
  - Still send the changes to the server
  - Relay the server’s state back to the user
  - If server gives different state, roll back!

This is actually the basis of “rollback netcode” - a popular term in the fighting game community.
Approach 2: Client-Side Prediction

In our client-server model, the server owns all world-state. But, we want the user to jump as they hit the button! An idea:

- Simulate the game on client-side as well
- Still send the changes to the server
  - Relay the server’s state back to the user
  - If server gives different state, roll back!

This is actually the basis of ”rollback netcode” - a popular term in the fighting game community.
In our client-server model, the server owns all world-state. But, we want the user to jump as they hit the button! An idea:

- Simulate the game on client-side as well
- Still send the changes to the server
- Relay the server’s state back to the user
- If server gives different state, roll back!

This is actually the basis of "rollback netcode" - a popular term in the fighting game community.
In our client-server model, the server owns all world-state. But, we want the user to jump as they hit the button! An idea:

- Simulate the game on client-side as well
- Still send the changes to the server
- Relay the server’s state back to the user
- If server gives different state, roll back!

This is actually the basis of "rollback netcode" - a popular term in the fighting game community.
Approach 3: Time Calculations

If we treat server-side as being the true state of the world, then in the client’s view:

- Their own character is slightly ahead in time
- Other users’ characters are slightly behind in time

So, one more approach is to calculate, based on ping, what the player was actually trying to interact with!

Side note: Do be mindful of how this affects player experience!
Addendum: Reducing Network Traffic
Reducing network traffic

We also want to reduce the network traffic generated by our game. This is for a few reasons:

- Frees resources for other computers
- Reduces number of network calls our computer makes
- Reduces stress on the network
Reducing network traffic

One good approach is interpolation:
- Only send/receive packets every other frame
- In the gaps, just interpolate between adjacent packets!

This resembles our client-side prediction approach to dealing with latency.

**Figure 11:** Interpolation of 2 points
References

- https://www.youtube.com/watch?v=aYX5TXMArTs
- https://en.wikipedia.org/wiki/Multiplayer_video_game
- https://www.britannica.com/topic/PLATO-education-system
- https://www.w3.org/People/Frystyk/thesis/TcpIp.html
This presentation was from the **Mines Linux Users Group**. A mostly-complete archive of our presentations can be found online at [https://lug.mines.edu](https://lug.mines.edu).

Individual authors may have certain copyright or licensing restrictions on their presentations. Please be certain to contact the original author to obtain permission to reuse or distribute these slides.