

ENSE 472: Digital Networks Netcode

University of Regina
December 5th
Fall Semester 2019

Jeremy Cross 200319513
Taylen Jones 200354271
Daris Lychuk 200361245
Kegan Lavoy 200378170

Table of Contents

1.0 Introduction.....	2
1.1 Problem Definition.....	2
1.2 Motivation behind the topic.....	5
2.0 Network Models.....	6
2.1 Dedicated Servers.....	6
2.2 Client Hosted.....	7
2.3 Peer-to-Peer.....	8
2.3.1 Delay-Based.....	9
2.3.2 Rollback.....	10
2.3.3 GGPO.....	10
3.0 Benefits and Disadvantages of Network Models.....	11
3.1 Benefits of Dedicated Servers.....	11
3.2 Disadvantages of Dedicated Servers.....	12
3.3 Benefits of Client Hosted.....	12
3.4 Disadvantages of Client Hosted.....	13
3.5 Benefits of Peer-to-Peer.....	14
3.6 Disadvantages of Peer-to-Peer.....	14
4.0 Challenges with Building Netcode.....	14
4.1 Lag Compensation.....	15
4.2 Hit Registration.....	15
4.3 De-synchronization.....	16
5.0 Case Study.....	16
5.1 Call of Duty Modern Warfare (2019).....	16

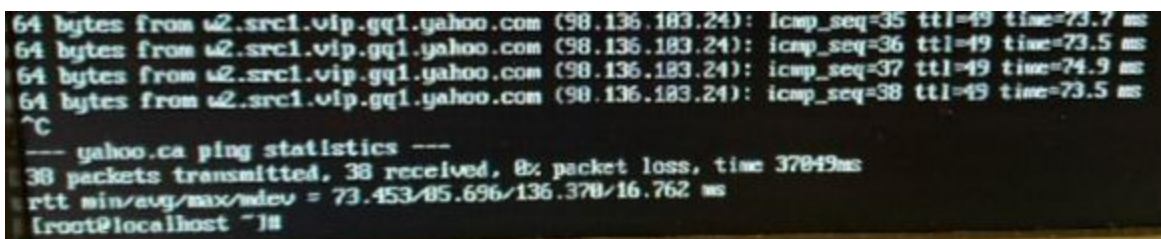
1.0 Introduction

For our ENSE 472 project we will be researching gaming netcode. Gaming netcode is another term used for networking for video games. This topic relates heavily on networking issues, from ping rates, to routing, to potential packet loss and so forth. We will cover different types of netcode like dedicated servers, client hosted, and peer-to-peer. We will explore how certain games utilize different netcode, the benefits and disadvantages to each, and issues that can still be improved upon today. We will look at the process of implementing netcode and the challenges of designing strong netcode for games.

1.1 Problem Definition

Networking does not come without some challenges. There are some major networking challenges when playing online multiplayer games. Some of the major challenges facing the network presented are: ping, routing, packet loss, update rates, and tick rates. These topics are all mainly related to processing capacity of the network. If the network you are playing a game on is incapable of adjusting or resolving the above issues mentioned, you will not have an enjoyable experience when playing any online multiplayer games. Netcode can in a way be classified as the foundation to every online multiplayer game on the market. If the foundation were to ever crack or break, nothing would be usable anymore, this is why netcode is so important. In order to have a better understanding of what netcode truly represents, the challenges this method is trying to solve will be broken down in order to differentiate “good” netcode, from “bad” netcode.

When talking about networking and how it relates to video games, the most general term that almost everyone, whether they play online games or not, has heard of is “ping”. Ping, also known as latency, is the time in which it takes to send a request to the server and then to properly receive the answer back. The higher the ping, the more delayed a response will be, making the game feel almost unresponsive. Pinging involves sending an Internet Control Message Protocol (ICMP) to the server with an echo request, with the server then responding back with an echo reply. The time taken for the echo request and reply is added together and this is your ping rate. As shown below in *Figure 1*, when pinging yahoo.ca, it will respond back with a response time from each ping request. *Figure 2* below also demonstrates how the calculation is made with a simple illustration.

A terminal window showing the output of a ping command to yahoo.ca. The output displays four individual ping results, each showing 64 bytes from a specific source IP (98.136.183.24) with increasing ICMP sequence numbers and response times. Below these, it shows a summary of the ping statistics, including the number of packets transmitted and received, packet loss percentage, and the round-trip time (RTT) in milliseconds (min/avg/max/mdev).

```
64 bytes from u2.src1.vip.gq1.yahoo.com (98.136.183.24): icmp_seq=35 ttl=49 time=73.7 ms
64 bytes from u2.src1.vip.gq1.yahoo.com (98.136.183.24): icmp_seq=36 ttl=49 time=73.5 ms
64 bytes from u2.src1.vip.gq1.yahoo.com (98.136.183.24): icmp_seq=37 ttl=49 time=74.9 ms
64 bytes from u2.src1.vip.gq1.yahoo.com (98.136.183.24): icmp_seq=38 ttl=49 time=73.5 ms
^C
--- yahoo.ca ping statistics ---
38 packets transmitted, 38 received, 0% packet loss, time 37849ms
rtt min/avg/max/mdev = 73.453/85.696/136.378/16.762 ms
[root@localhost ~]#
```

Figure 1: Ping response from yahoo.ca



Figure 2: Ping calculation [1]

Routing involves how the data packets sent from your local machine are directed to the server properly. The greater the distance between the machine and the player, the more server hops the packet will have to take, and therefore the greater time needed to communicate. When looking for the best route to take, your machine will look at how many hops, and not necessarily the physical distance between your machine and the server as shown in *Figure 3*. Hops are considered over physical distance because your information has to follow the same path the line is taking, it can not magically communicate in a straight line. Routing not only deals with the distance between the machine and the player, but it also has to resolve congestion and flow control problems obtained when communicating.

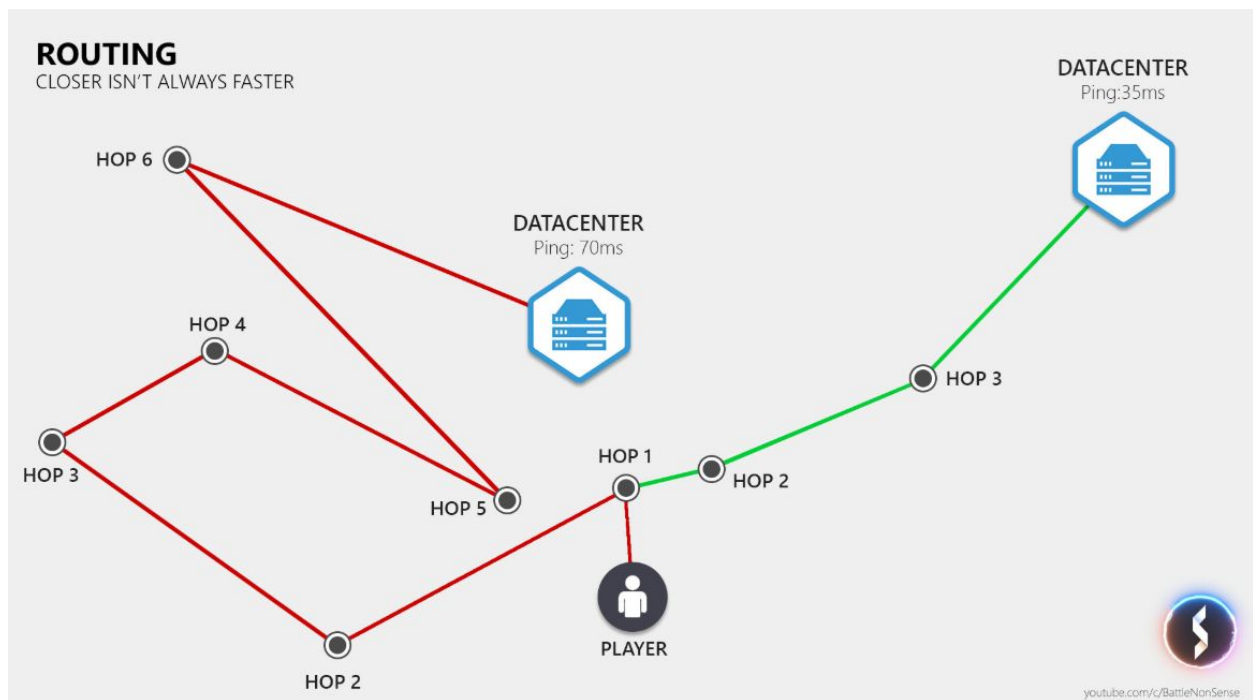


Figure 3: Routing communication comparing hops [2]

Packet loss describes the issue of your data properly reaching its location or not. If a packet is lost, it can cause major trouble for real-time applications, such as online video games. Packet loss can also be intertwined with routing in a sense, as the more hops a packet needs to take,

the greater the chance of packet loss as well. As mentioned previously, packet loss occurs for multiple reasons, however, one of the reasons being congestion in the network. If the network is too congested, clients and servers will be unable to communicate properly, and packet loss will occur. Another major reason could be due to hardware or software issues such as broken ports, cables, or drivers. In *Figure 4* below, other causations of packet loss are represented.

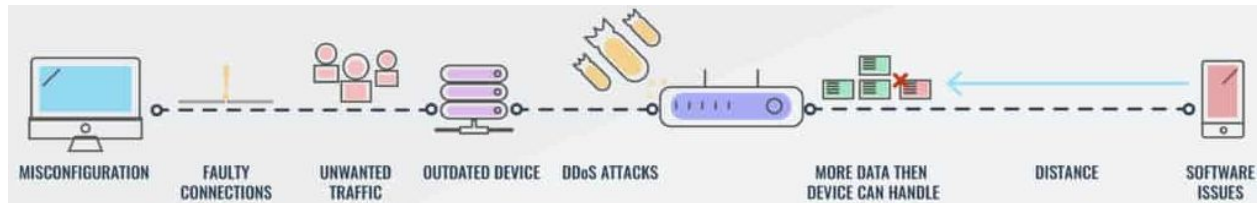


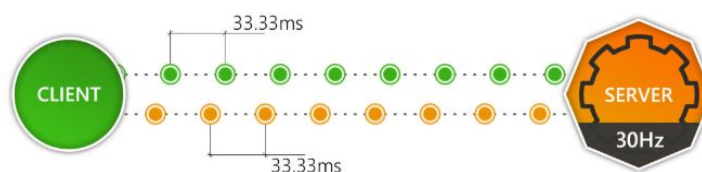
Figure 4: Causation of packet loss [3]

Update rates is a term used to describe how often a game sends and receives data between its client and server. The greater the update rate, the less delay added on top of the time it takes for the data to travel, as well as packet loss becoming less concerning. As shown in *Figure 5*, changing the frequency of the update rate, can either partially reduce or increase delay. When decreasing the update rate issues arise. The less amount of packets received, the more important the packets being received are. To try and minimize the importance of each individual packet, update rates are increased. This is so that if a packet were to be lost or dropped, as previously mentioned, it would not have a drastic impact on the players/clients.

UPDATE RATES

ADDITIONAL DELAY

30Hz UP & DOWN:



60Hz UP & DOWN:



youtube.com/c/BattleNonSense

Figure 5: How frequency of update rates changes delay [4]

Tick rates, also known as simulation rates, relates to how often the server actually processes and produces all of the data received from the clients in order to actually simulate its proper response. When a tick starts, the server receives the data and begins to process it as shown in

Figure 6. The response to be sent out has to be processed within a certain time frame in order for the game to run smoothest and with the least amount of delay. Servers are required to process the simulations as fast as possible. If the server were to miss its given processing window time frame, clients would experience multiple issues such as teleporting, rubber banding, physics failing, or hit rejection.

TICK RATE:

Simulation, Tick Processing

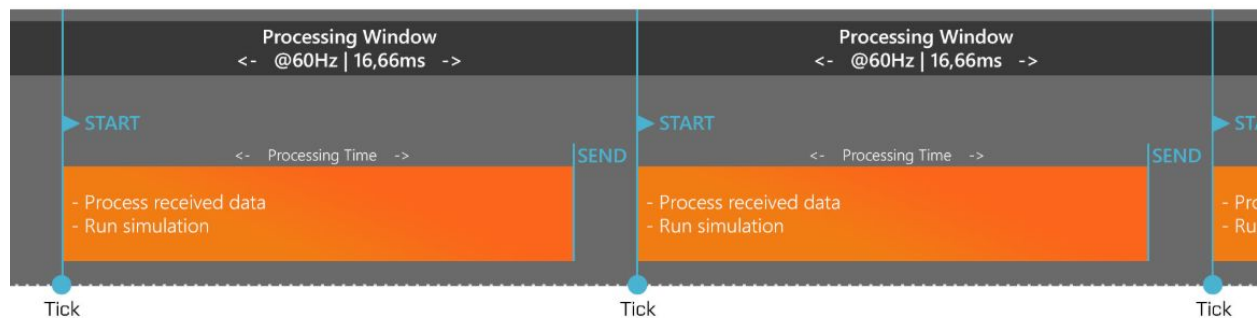


Figure 6: Tick rate processing [5]

The big picture of all of these challenges is that video games would not be able to be played online if it were not for netcode. There needs to be some sort of synchronization between clients and servers for it to be a usable application. All modern servers need some form of lag compensation if they want people to enjoy their experience. Lag compensation involves everything previously mentioned relating to delayed data sent and received between the client and the server. By combining all the knowledge from the previously broken down topics, the server is able to compensate for their clients so that everything will still feel responsive and without delay. These issues that a user may face all link together and are brought upon individuals due to the major challenges mentioned. They are all linked to the communication between networks which involve the transport layer, layer 4, in the ISO/OSI Model. Which type of network transportation would be better for the communication between clients and servers for online video games? This is dependent on the software developers or the gaming company. TCP and UDP both have their positives and negatives when dealing with netcode. TCP has a high overhead cost and increased latency, yet it is able to handle error conditions and is simpler than UDP. UDP ends up having lower overhead cost as well as decreased latency compared to TCP. UDP automatically seems better, but may unfortunately need networking code to be implemented into the game engine in order to handle the error conditions TCP handles by itself. However, by implementing networking code into the game engine increases the complexity of the engine and possibly brings forth new issues. This is a company based decision that needs to be made when creating an online multiplayer game.

1.2 Motivation

This topic was chosen for many reasons, however, the primary reason being that it is used in everyday life by millions, if not billions of people, and most do not know about it. Being that netcode is used in every online multiplayer video game on the market to this date, it is a fairly important topic. Being that there is no competition to netcode implemented, or even being discussed currently, we believe that netcode will be a technology that will continue to advance into the future. Even though the process of netcode can be considered a monopoly process, every software development company uses the netcode standard to fit their needs. This is an interesting topic to dive into to see how everything is really happening when you are playing an online multiplayer game.

2.0 Network Models

This section will discuss the various different network models that are used with games and describe how they function. There are three general network models that are used for gaming netcode. The network models are dedicated game servers, client hosted or listen servers, and peer-to-peer. The different common variations of peer-to-peer based networks are delay-based and rollback netcode. These models will also be discussed in this report. Lastly, this section will discuss different genres of games that are used with these different network models.

2.1 Dedicated Servers

With this network model clients or players connect to a dedicated server. A dedicated server means that a specific computer or server is utilized for the main purpose of hosting other players. These types of servers provide enough bandwidth and low latency for everyone connected. This type of network model is ideal for online team-based games with several different players participating at the same time. Types of genres that reflect this are first or third person shooters (FPS or 3rdPS), multiplayer online battle arenas (MOBA's) and massive multiplayer online games (MMO's).

There are a few different variations for hosting dedicated servers. These consist of cloud hosting, and in-house or off-site servers. Cloud hosting consists of having a server reside on a computer provided by a service provider. In-house servers are on site, require manual maintenance, and a large enough space to house the hardware and equipment. *Figure 7* illustrates the concept of the dedicated server model.

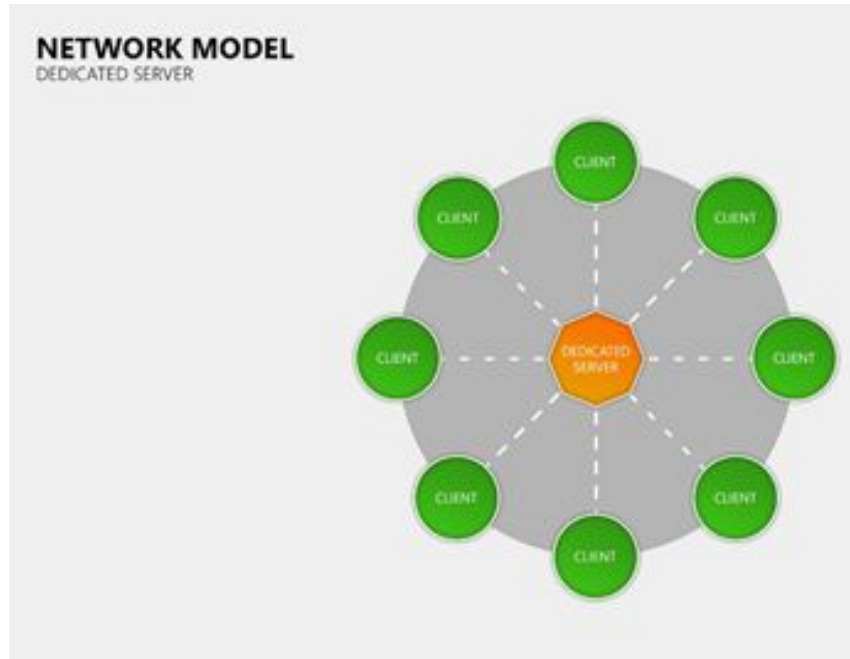


Figure 7: Dedicated Server Network Model [6]

2.2 Client hosted

This type of network model is similar to the dedicated server model except a client or player acts as the host. This is known as client hosted or a listen server. The player that is usually hosting however has the advantage because they have little to no lag while the remaining players experience more lag. The host player is able to gain advantage by being able to see other players before they do. The quality of the connection is entirely dependent on the host connection. Host migration can also occur if the current host decides to leave and another host is elected to become the new host. This will cause the game to pause while the new host is selected and the migration is in process. This type of network model is also used for a wide variety of team based games that consist of several different players participating at once.

Figure 8 illustrates the concept of the client hosted model.

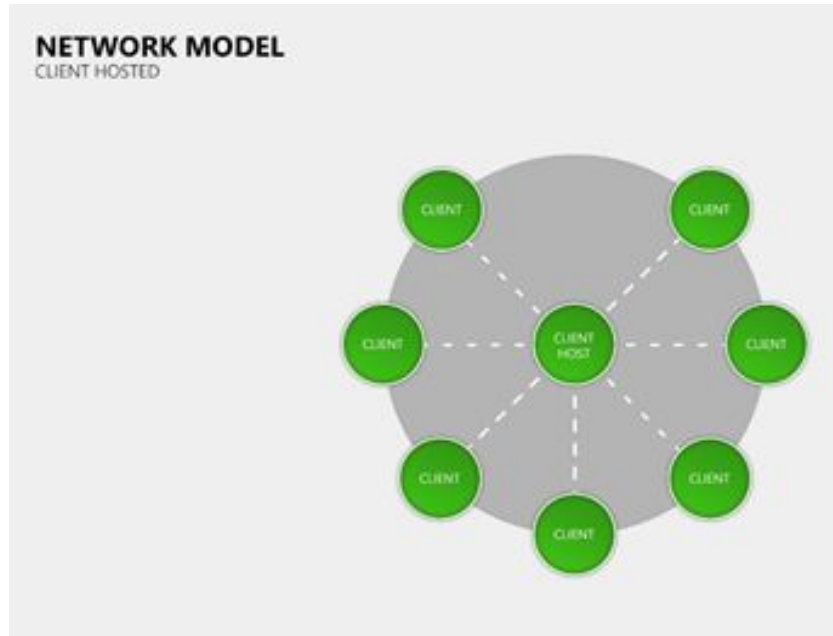


Figure 8: Client Hosted Network Model [7]

2.3 Peer-to-Peer

This type of network model is generally utilized for 1v1 based games although some games with several players also use this model as well. With the peer-to-peer model, clients or players directly communicate with each other. There are two commonly used variations of this network model, the first is delay-based netcode and the other is rollback netcode. The most common games that utilizes delay-based netcode and rollback netcode is the fighting game genre. Fighting games require fast input speeds and good communication across the network is critical to having a pleasant or quality experience. For a proper online experience while playing a fighting game, the game should function at 60 frames per second. Each frame is comprised of 16 milliseconds and considers numerous information for each frame. This information consists of running CPU AI, animating character moves, and detecting hits and inputs. The time it takes for one player on one machine to receive an input from another player on another machine online is based on the ping. If the ping is high then the amount of time the input takes to travel to the other player will be longer. If the game is offline and two players are playing on the same machine then the correct inputs are processed as expected without any delays and are displayed on the same screen for both players to see. Games use lockstep networking to send information to each other in sync to make sure the game states remain the same. If the game states from both players begin to disagree about each other then the game is desynced or abandoned. Lockstep networking provides a good anti-cheat solution. This is because if one player is using some form of in-game modifications, the two games will not match with each other and be desynced. The methods of dealing with input delay are handled with delay-based and rollback netcode as described below in the following sections. *Figure 9* illustrates the concept of the peer-to-peer model.

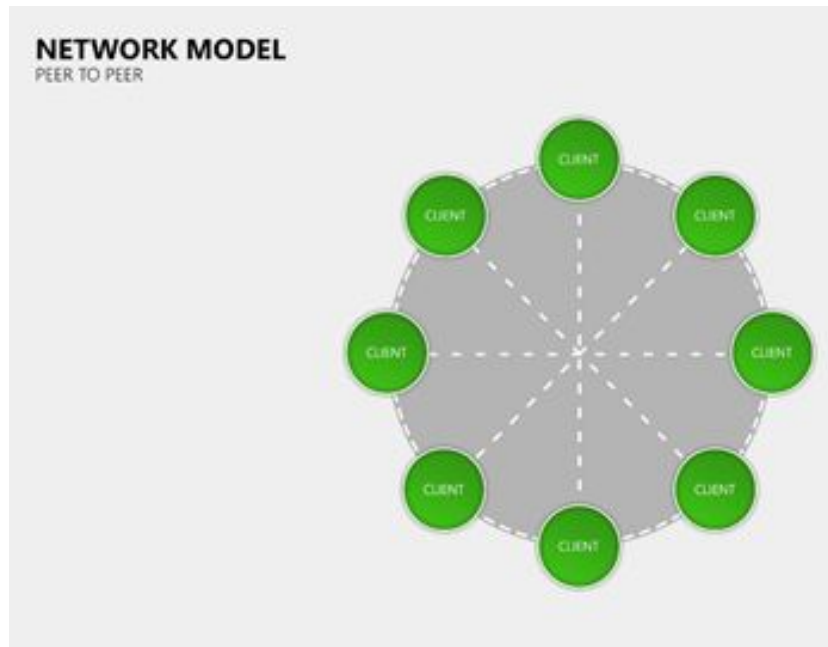


Figure 9: Peer-to-Peer Network Model [8]

2.3.1 Delay-Based

Delay-based netcode is the simplest and most common netcode to implement using lockstep networking. This form of netcode works by creating artificial delay for the player on the local side to allow the input from the opposing player to catch up. The delay is equivalent to the amount of time the opponent's inputs take. This allows both inputs to arrive at the same time and allows them to be executed on the same frame. If the frame delay remains rather consistent due to the distance between both players being rather short where the ping is relatively low, frame delays can seem relatively unnoticeable. However, if the distance between both players is rather far or the ping is inconsistent then the quality of the game is quite unplayable. The distance between the two players determines the amount of delay that is required. If the distance is far then more artificial delay is required because the network travel time is greater. If the distance is short then less artificial delay is required. Delay-based netcode is still commonly used in fighting games today mainly because of its simple implementation and cheap cost. The next section will discuss a different approach for designing netcode for 1v1 games with rollback netcode.

2.3.2 Rollback

The way rollback netcode works is that it never has to wait for inputs like the delay-based model. The rollback model continuously runs the game normally. The local players inputs will be displayed on the screen normally while the opposing players inputs will arrive several frames later. The game will rollback or rewind to the original frame that the opposing input was meant to be applied on and re-simulates the frames ahead to reach the present frame. To the local

player, the opposing players animations may appear several frames ahead already. This also temporally breaks lockstep model because the game state shown to both players is different. With this rollback model inputs are not lost like the delay-based model. If any rollbacks are required then the inputs are re-applied.

The rollback model extends the idea of rollback with predicting what the opposing player will do next. If there are missing frames then the rollback model will duplicate the input from the last known frame across the unknown frames. When the correct inputs do arrive during the frames ahead the game must determine a course of action based on whether they match the predictions or not. If the inputs match the predicted inputs then no rollback is required and the games state continues on. If they do not then rollback is required and the game rewinds to the last correct game state and the correct inputs are applied.

The ideas from both models can be applied together where the rollback can be built on a delay-based framework. This combine the properties of both together. This reduces the amount of times rollback is required. Some games allow the player to be able to set the frame delay before rollback occurs. This means that rollback will only occur with inputs when the frame delay exceeds the set limit by the player.

In order for any netcode solution to work the games states need to be completely in sync. Simply by locking the frame rate a 60fps is not enough to ensure that the beginning of the game up until the end of the game will remain in sync. Many other issues outside of networking could put the games states out of sync. To solve this problem rollback systems will pause the player that is ahead one or two frames to allow the other player to catch up. This will prevent the games from moving to far apart from one another.

Overall the rollback model is a better improvement over the delay-based model. The local players inputs will always be immediately processed to the screen and any rollback that may be required will re-apply the input to the correct frame. There is no way that inputs can be lost from network lag like in the delay-based model. Delay can seem rather invisible to most players due to the rollback method and produce no visible changes to the players. However, the difficulties with incorporating rollback netcode into games is that rollback has to be considered at the start of development and the game has to be built with rollback already in mind. In order for the rollback to work the game has to save and load different game states in the background. This is called serialization and this can be very time-consuming operation. Changing the way that data is stored for this for serialization to work well could change how some of the core systems work within the game.

2.3.3 GGPO

GGPO is a free open source rollback library. The function of this library is to keep the games synced and handle game states. GGPO will tell when games are out of sync and by how much. It will keep track of inputs from the players and understand when the games need to rollback and by how much. If there is rollback then it will apply any new inputs that are required.

However, it still falls on the developers to configure their game to work with the GGPO framework. GGPO only tells the when and why to rollback as well and the inputs to apply with the rollback. Some developer may choose to create their own in house rollback netcode that handles game sync and game state.

3.0 Benefits and Disadvantages of Network Models

3.1 Advantages of Dedicated Servers

If you have ever played an online video game, you have probably heard the term dedicated server used. Dedicated servers are among the most popular of choices for hosting online games in today's age, offering a long list of advantages compared to other methods. One of the most important benefits is the reliability and accessibility that dedicated servers offer. As opposed to shared hosting, where a client would potentially be sharing the server resources with other sites using it as a host, a dedicated server will give them 100% of its power without the worry of a potential bottleneck. This in turn leads to more scalability, if a client ever needs more resources due to server load they will be available. With respect to the aspect of accessibility, a dedicated server offers its clients all administrative access so that they may take a more hands on approach to programming it, and potentially install any programs they wish. They allow the client to have complete control on who is allowed into the server, what rules are in place, and kick players who may be slowing down the system with high ping.

Dedicated servers also offer a very high level of security. The infrastructure within a server includes high level monitoring as well as firewalls. Unlike virtual servers, dedicated servers do not reside on a machine with multiple data ports open. And since the server is not shared, the address of the machine can be controlled. Another aspect of high security on dedicated servers is the actual physical location of the server. Dedicated servers are almost always stored in highly secure areas, with access only to company personnel.

One of the most successful games to implement dedicated servers is Minecraft. Minecraft allows users to download their server software directly from their website, and easily set up their own private server. From here the user can set rules, permissions, and play on their own private server without the worry of other players entering and ruining their gaming experience.

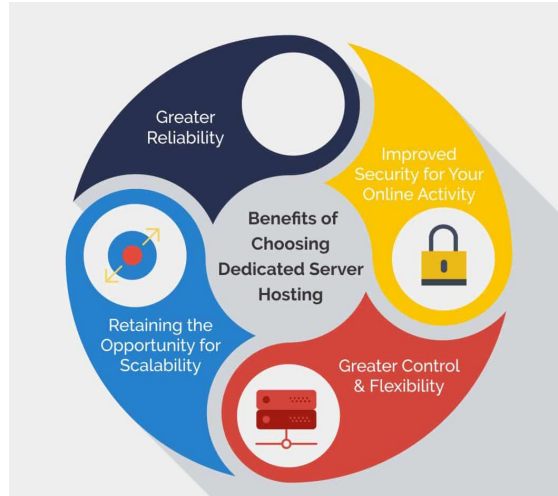


Figure 10: Benefits of choosing a dedicated server network model [9]

3.2 Disadvantages of Dedicated Servers

When it comes to online games with more than two players, dedicated servers are almost always the best choice. However, this doesn't mean that there are no disadvantages to using dedicated servers. One disadvantage of using a designated server netcode strategy is that it is very expensive to implement. The more online players that need a server to connect and play the game, the number of servers needed goes up. This means for games that have millions of players such as call of duty or world of warcraft, the number of servers needed can be immense and therefore expensive. This forces companies to either make their players pay for the servers through a subscription type model like world of warcraft, or pay for the servers out of their own pockets.

Another disadvantage that dedicated servers have is accessibility. The amount of latency that a player has is largely connected to how far away they are located from the server they are connected to. This means that certain regions may have much greater latency than others simply because the servers are farther away. This again puts some pressure on companies to make certain decisions. They can either add more servers in more regions which will increase their costs, or potentially risk their player base becoming unhappy and leaving the game behind entirely.

3.3 Advantages of Client Hosted

Client hosted servers were the next step in netcode, succeeding peer-to-peer as the go to method. The apparent limitations in P2P was the driving motivation behind client hosted development. Most of the benefits of using this approach fall under the same category as using a dedicated server, but without the cost due to a client acting as the host of the game. This might seem like an obvious choice over using a dedicated server, since it comes with the same

advantages with no cost, but there are many more drawbacks to be discussed later in this report.

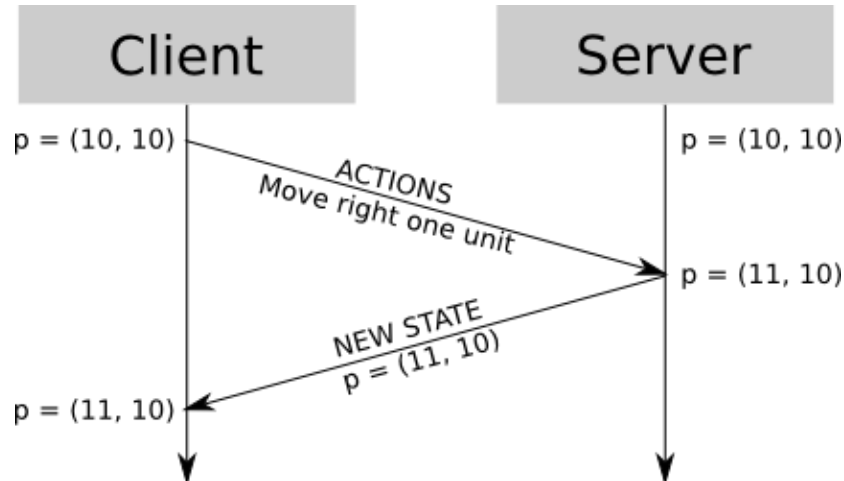


Figure 11: Basic Client Server interaction [10]

3.4 Disadvantages of Client hosted

A variety of issues can arise when a client or user of the game is allowed to become the host. Since the host is a player within the game, that particular player has an immense advantage over the other players because they will essentially have zero latency while the other players will not. This allows the host to see everything in the game in real-time while the other players have a delay, causing an unfair match. This advantage can be amplified by the host using something known as a “lag switch”. A lag switch is when the host artificially increases the latency of other players while keeping their own near zero. This causes the other players to experience an unplayable amount of delay while the host has next to none.

Another issue that can occur in a client-hosted netcode is the host's internet connection being overwhelmed by the amount of traffic on the network. In a client-hosted netcode, all other players connect to the game through the host's internet connection. If the host's internet speed is too low or is using something such as wifi to host the game, the amount of traffic generated by all of the other players will congest the network and the game will once again become unplayable for the players due to high delays and packet loss.

A fourth issue when using a client-hosted netcode is known as “host migration”. Host migration occurs when the player that was hosting the game quits or the connection to the game is lost. Since no one is hosting the game when this happens, the game must pause and choose a new player to become the host. Once a new host is chosen all of the players must now reconnect to the new host before the game can continue. This can be very annoying to players trying to play the game and it may cause them to lose interest in playing.

One final issue with client-hosted netcode is that the host is able to see the IP addresses of the other players in the game. This poses a very high security risk because it makes the host able to perform things like DDOS attacks against the other players if they have the knowledge. This essentially allows the host of the game the ability to cheat and gain a significant advantage over others while creating a very unsatisfying and frustrating experience for others.

3.5 Advantages of Peer-to-Peer

With peer-to-peer netcodes being one of the earliest forms, it can be difficult to justify using it over one of the previously aforementioned methods, but there can be some benefits to using this old school technology. The biggest difference between peer-to-peer and client hosted servers is all the computations are done on the server, with the clients sending the information across. In a peer to peer network, there is less data to be sent across, which can in turn be big in respect to saving time.

Peer-to-peer networks are also substantially cheaper, and are typically a good choice for indie mobile game developers who have a limited budget to spend on hosting services such as dedicated servers. Servers often require expensive hardware, equipment and regular maintenance, where P2P networks only require a connection between the two machines to connect and have functionality. But this all comes at the cost of reliability, and unless both parties on the P2P network have stable, high speed connections, servers are almost always the best route to take.

A good example of games that use these features to their advantage are real time strategy games. Frequently these types of games will have thousands of units on screen at any given time, which would be too large to effectively communicate between players. Instead they send the associated commands of the units back and forth. Even with this, peer-to-peer is very rare to find being implemented into modern netcode due to major security risks and other disadvantages to be discussed later in this report.

3.6 Disadvantages of Peer-to-Peer

One issue that is similar to that of a client-hosted netcode in a peer-to-peer netcode is that since all of the players directly communicate with all other players in the game, all players are able to see the WAN IP's of the other players which is a security risk for DDOS and other types of attacks. Since the server is not controlled by the developers, but shared and hosted by the players, another issue that arises is the developers ability to stop cheating. In a dedicated server netcode model the developers are able to monitor the server and ban or deal with players who are attempting to cheat. In the peer-to-peer model however, it is much more difficult for the developers to monitor the games and players activities.

A third issue associated with the peer-to-peer netcode model is having more than two players heavily increases traffic on a players internet connection. The increase in traffic can cause increased latency and delays to the game as well as affect other devices connected to the same internet connection. With too many players connected using the peer-to-peer model, the traffic soon becomes overwhelming for the vast majority of users home internet connections and the game becomes unplayable.

4.0 Challenges with Building Netcode

In the context of multiplayer online games, There are obviously many challenges that netcodes need to deal with in order to allow the players to join a network and play together at the same

time. Some important topics that netcodes take into consideration are lag compensation, hit registration, and de-syncing. Without netcodes dealing with these issues, players would be able to connect to the game with other players, but due to latency and other factors would almost never be able to hit or see other players. We will take a more detailed look into each of these topics and see how netcodes handle these issues.

4.1 Lag Compensation

All players experience at least some form of latency and delay when playing multiplayer online games. This means that although one player may see an enemy at a location, the server may see the enemy at a different position, and the enemy may see themselves at a different position as well as shown in *Figure 12*. If netcodes did not compensate for these delays at all, players would essentially always be shooting where an enemy was and not where they are, even though it may seem differently.



Figure 12: Viewpoints of single player [11]

Netcode tries to deal with this problem by compensating for a portion of the delays, allowing players to shoot at what they see and still hitting the enemy. However, it is possible to overcompensate these delays, so developers have to be careful. If too much of the delay is compensated to try and give players with high latency a good experience, it may ruin the experience for others. This is because they would be able to get shot while they are behind walls and cover by players with higher latency, due to those players still seeing them not behind the cover. Developers have recently started to set a threshold for lag compensation. This means that a certain amount of latency is compensated for, but if a player has too high of a ping, they will have to start manually leading their shots to account for their latency.

4.2 Hit Registration

Hit registration is the term used for determining if a player's bullet has hit something or not. Hit registration provides a different challenge depending on what netcode model is being used. For the client-hosted and peer-to-peer netcode models, hit registration is done by the player's game client which then communicates to the server telling it that the player hit something. This method is extremely accurate, but can also be easily hacked. The more common and secure method of

hit registration is to let the server handle it. The server determines which direction the player shot and then determines if it hit anything or not. This method is far more secure than client side hit registration, but is still hackable.

4.2 Desynchronization

One final challenge that netcodes need to deal with is de-sync or desynchronization issues. A de-sync issue is when 2 or more players are seeing multiple states of the game at the same time. For example, if one player sees a door as open and another player is seeing the door as closed. De-sync issues are usually caused by things unrelated to networking such as bugs within the game, but can still occur if things like the tick rate are too slow so developers still need to be aware of desynchronization issues on the networking side as well.

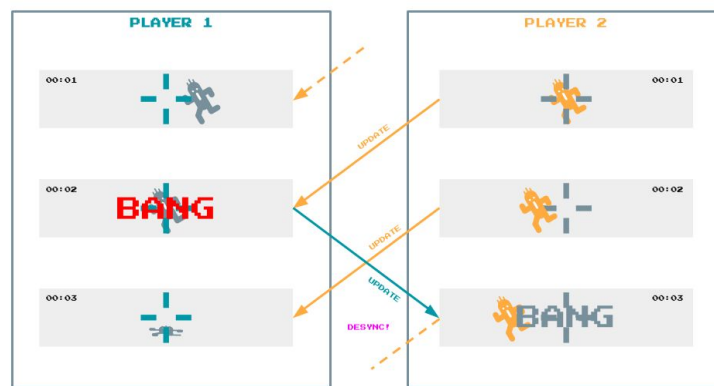


Figure 13: How Desynchronization can occur [12]

5.0 Case Study

Due to every online multiplayer video game using at least one of the three models of netcode, we will be doing case studies on individual games and how and why they chose the netcode model they use.

5.1 Call of Duty Modern Warfare (2019)

Call of Duty Modern Warfare was just recently released in 2019. To this current day it uses a client hosted or dedicated server network model, depending on the game mode. As previously mentioned, client hosted, is where game matches have one client host the server for many other clients in the match. Being that the game is not always running on a dedicated server, like most other big game titles, competitive gaming can prove to be somewhat challenging. By using the client hosting network model, tick rate, one of the problems netcode is trying to eliminate and/or reduce, becomes much lower. As previously mentioned, when tick rate becomes lower, the server will have less time to process and produce all of its data (30Hz vs. 60Hz). Another huge problem with using the client hosted network model means that there needs to be some sort of host migration. This means that if a client that was currently hosting the server disconnects, the game needs to be transferred to a different host client. By doing this, clients

unfortunately have to deal with more problems in routing, packet loss, etc. while the migration process is taking place. When acting as the host of the game/server however, the hosts latency, or travel time between host and client, becomes almost next to nothing. This is because there is only processing delay taken into consideration for the host being that they are the server as well. The only client able to benefit from client hosted matches is usually only the client themselves. When examining the data in *Figure 14*, you can see that the server update rate is only around 12Hz, which can cause major problems in delay and other issues.

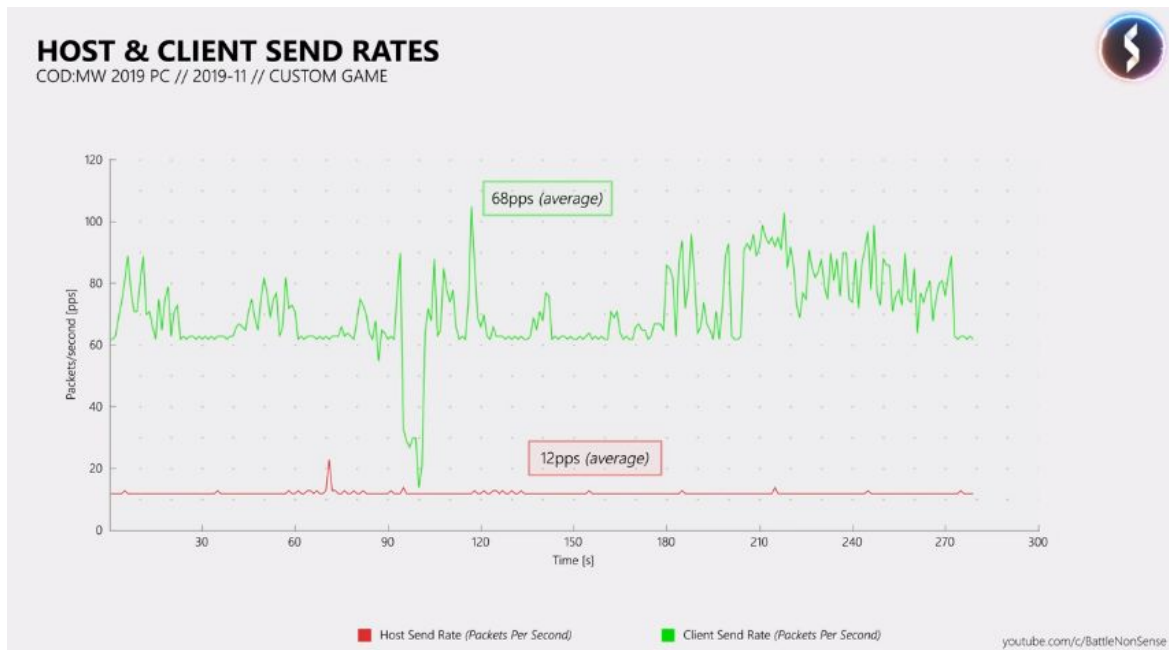


Figure 14: Client hosted network model used [13]

When examining the update rates between the dedicated server and clients, it appears normal, around 60Hz from the client and server, as shown in *Figure 15*. However, in certain game modes, the data sent from the server must be split into multiple packets as shown in *Figure 16*. If minor packet loss were to occur, you could lose an entire update causing lag issues. This can be extremely frustrating, luckily there is not very much data being sent from the server that has to be split into multiple packets. Most of the data being sent is controlled to fit fewer packets now.

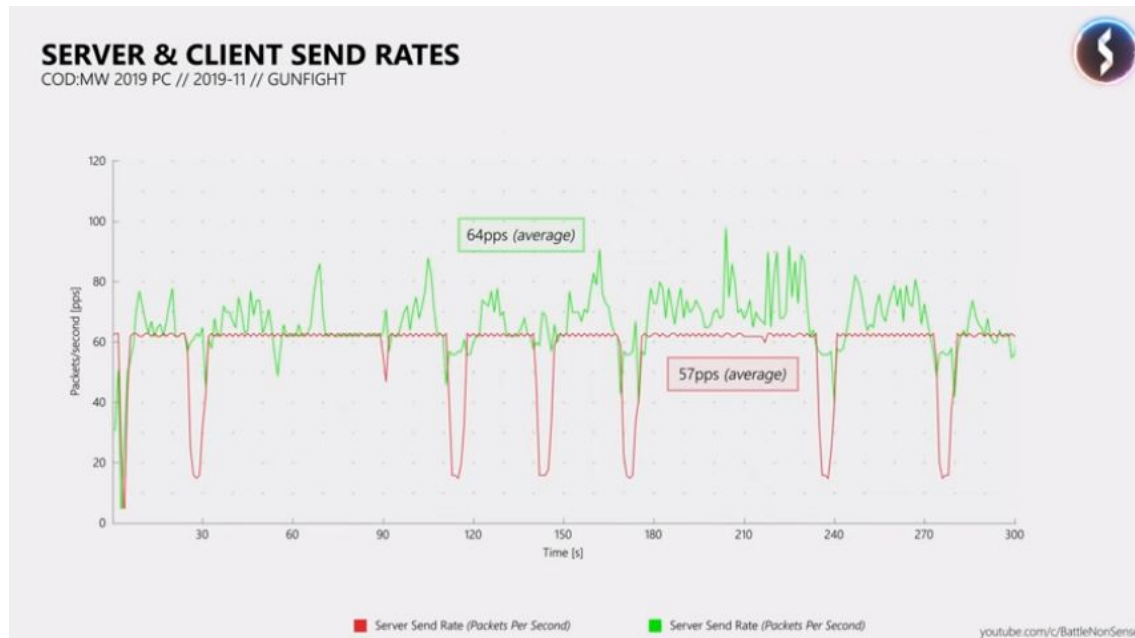


Figure 15: Server & Client Update Rates [13]

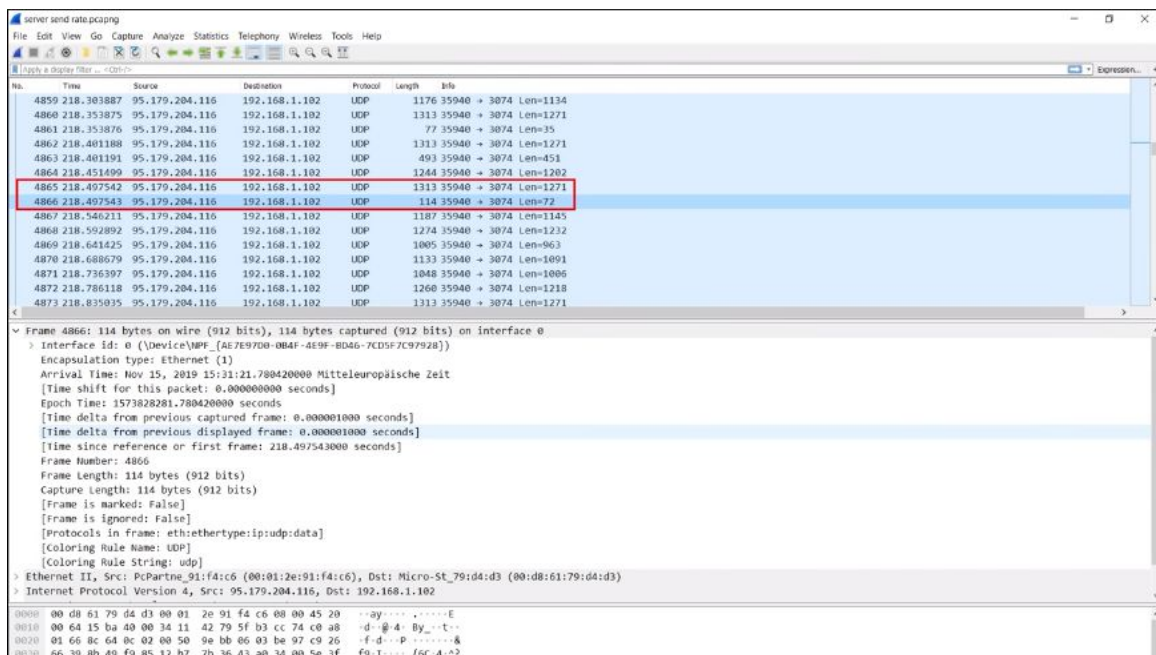


Figure 15: Splitting update data sent to multiple packets [13]

Being that there are always going to be network issues with games, Call of Duty Modern Warfare (2019) netcode decisions seem to be handling clients fairly well. Some may complain that custom matches/games should be running on dedicated servers just like other major titles currently are, however, there are of course disadvantages to using the dedicated server network model.

References

- <https://www.pcgamer.com/netcode-explained/>
<https://phoenixnap.com/blog/what-is-a-dedicated-server>
<https://phoenixnap.com/blog/dedicated-server-benefits>
<https://arstechnica.com/gaming/2019/10/explaining-how-fighting-games-use-delay-based-and-rollback-netcode/>
<https://cdn.mos.cms.futurecdn.net/yWELosJ4Yz2893PX7BCkhj-650-80.gif> [1]
<https://cdn.mos.cms.futurecdn.net/G7Y4am8yDSHXkiUbJmULdj.jpg> [2]
<https://cdn.comparitech.com/wp-content/uploads/2019/04/Causes-of-packet-loss-01-01-01-1024x427.jpg> [3]
<https://cdn.mos.cms.futurecdn.net/Nk3CBE58zw5Xwx34JpKwdj.jpg> [4]
<https://cdn.mos.cms.futurecdn.net/y4MQk9zMFu67LBUMPDCYj.jpg> [5]
<https://www.accuwebhosting.com/blog/benefits-dedicated-hosting/>
<https://cdn.mos.cms.futurecdn.net/hfaqxreyWd52xQzybDCeXj.jpg> [6]
<https://cdn.mos.cms.futurecdn.net/EdYB6W8fyz3Y6CyBFEntXj.jpg> [7]
<https://cdn.mos.cms.futurecdn.net/CXE9TpCDXLcJwkBK4h5hXj.jpg> [8]
<https://cdn.mos.cms.futurecdn.net/Ru7TeygLc9G3TwFmFZ68sB.jpg> [11]
<http://www.hosting-promo.com/wp-content/uploads/2017/03/benefits-of-dedicated-server-800x710.jpg> [9]
<https://www.gabrielgambetta.com/client-server-game-architecture.html> [12]
<https://medium.com/@meseta/netcode-concepts-part-2-topology-ad64f9f8f1e6> [12]
<https://www.youtube.com/watch?v=eo3Bh69xmoQ&t=83s> [13]