Extended Essay

Computer Science

Title

Evaluation of Blockchain as Replacement for Client-Server Model for Asset Management in Online Games

Research question

To what extent is a blockchain-based system a viable option for the management of assets of an online game involving inter-player asset trading?

Session

May 2021

Word count

3,842

Table of Contents

I. Introduction

A rise of incidents regarding information security has been seen with the rise of the internet. It also indicates more vulnerabilities of users, making online theft of resources more and more common. For larger corporations being affected by hacking attacks, this has been disastrous in the past, taking the example of the 2016 Bangladesh Bank cyber heist, wherein over $102 million were stolen, some of which was never recovered. Similar large-scale theft has been done with trading, wherein assets are exchanged amongst players daily. Therefore, cybersecurity and cryptography have become essential for daily functioning, and one well-known implementation of this principle of cryptography over networking that has seen a rise in popularity over the years is blockchain technology.

Asset trading games involve assets which have a specific value, use, and fixed quantity. This value system causes monetary compensation, with individual collectors paying several thousands of dollars to acquire such rare assets. However, during transactions, these assets have been known to be stolen along with the actual currency paid for it. Thus, game companies have been successfully sued in the past for compensation for these assets. An effective solution for this dilemma would be the use of blockchains for asset management and exchange. The blockchain would allow for complete transparency concerning the exchange of assets, and the immutability associated with blockchain systems would prevent robberies of such assets. This essay investigates the feasibility of a blockchain system in the management of such assets in games involving inter-player asset exchange.

II. Methodology of Evaluation

This research paper evaluates the feasibility of blockchain as a system to manage assets in games involving inter-player asset exchange. In order for achieving this, a framework for testing has been defined to provide for fair and unbiased evaluation. The framework acts as a qualitative method to measure the suitability of the framework for the concerned tasks.

II.1. General Criteria

Level	Description
3	The criterion objective is fully met and satisfied for suitability in asset management. The implementation meets all requirements described, lending itself to a secure and robust protocol for asset trading in this context.
2	The objectives outlined are met to an extent allowing for adequate functionality in the context of asset management. Minor problems can be worked around to meet the criteria.
1	Lapses in the framework exist, preventing parts of the specific criteria from being met. Some aspects of the criterion are still met and can benefit asset trading and management.
0	The framework under evaluation does not meet the criterion in any way. The significant lapses present in this criterion affect its use for asset management.

Figure 1. General criteria for analysis

II.2. Specific criteria for success

II.2.1. Speed and reliability

This criterion focuses on the real-world performance of the frameworks, and what it means for asset management.

Criterion	Description
Request speed and protocol overhead	Concerns the method of data transfer across the framework, and its appropriateness for frequent transfers of multiple assets, as is with simultaneous asset trading solutions.
Device-side processing	Regards processing request overheads on the server-side made on the framework. It includes network protocol considerations.

Figure 2. Criteria for analysis under speed and reliability

II.2.2. Storage

This criterion focuses on the framework storage requirements and targets possible data faults.

Criterion	Description
Integrity of transaction data	It relates to any measures in place preventing corruption of records of transactions made using assets through the concerned framework.
Integrity of asset data	This relates to any measures in place preventing corruption of assets being managed and transmitted through the concerned framework.
Loss prevention and tracing	Regards the effectiveness of safeguards put into place to handle or prevent incidents of mishandling/corruption of data.

Figure 3. Criteria for analysis under storage

II.2.3. Computational power

This criterion focuses on the framework computation requirements to facilitate asset management and trading, and possible effects on player system requirements.

Criterion	Description
Client-side processing and requirements	Regards the system requirements that stakeholders must meet to access the framework for completing a transaction.
Framework processing and requirements	Regards the performance of the framework and computational requirements to maintain a running state.
Network connectivity and reliability	Concerns the reliance on the framework on network connectivity when handling tasks or transactions.

Figure 4. Criteria for analysis under computational power

III. Client-server models

The client-server model is a distributed application architecture partitioning workload between two categories of stakeholders: the server providing a resource or service, and the client requesting the resource or service over a computer network. A given system can act as both client and server in this model. Server programs started by the host share resources with the client, but clients do not share resources with the server. Clients merely request services communicated through TCP based protocols.

III.1. Topology design

Different client-server topologies can be implemented in different programs. Server-hosts can communicate with one another for sharing of resources. A single server can push resources to multiple clients, and a single client can request or pull resources from multiple servers. As a result, servers often have higher hardware requirements than clients.

Clients can be classified into thick clients and thin clients. Thick clients do not consume any of the server's resources to execute applications. These clients suffice the hardware requirements for running programs pulled from servers. Servers only provide the data necessary to run these programs, run independently by the client.

Thin clients depend heavily on the server's resources for executing a program. They are relatively low-performance terminals sharing resources from the server to run applications

hosted by the server. These clients cost less due to less hardware redundancy for running programs pulled from a server.
Servers are also categorised into iterative and concurrent servers based on how they process client requests. Iterative servers process client requests through TCP/IP one at a time, causing unnecessary delays not attributable to latency. Concurrent servers process multiple requests at once through subdivision of loads into sub-processes. These are better suited for topologies where multiple clients are dependent on a single server.

Client-server architectures, in real-world applications, typically rely on multi-tier architectures. These physically separate different server functions into processes run by groups of servers, known as layers of the server architecture. Tier segregation allows for better resource management between servers and provides flexibility should a layer be added, removed, or modified. It also promotes redundancy and reduces the load on servers that process the backend, allowing for future expansion.

III.2. Anatomy of a client-server transaction (Steam)

The Steam in-game or inter-player transaction system relies on HTTP requests to three tiers of servers, secured using SSL v3 128-bit protection. These HTTP requests pass through three layers of servers (game purchasing servers, Steam billing server, Steam client) before the transaction is approved. The SteamWorks API by Valve Corporation is utilised for transaction approval.

Stage	Description
Initialisation	The SteamWorks API is initialised before the transaction begins. This is necessary to communicate with the Steam billing servers and requires a header file, lib file, and DLL to continue.
Request bundling	The user's Steam ID (64-bit unique identifier), ISO 3166 country code, ISO 4217 currency code, and ISO 639-1 language code are bundled with regular purchasing data. This is obtained through the *ISteamUser::GetSteamID, ISteamUser::GetCurrentGameLanguage,* and *ISteamMicroTxn/GetUserInfo* API calls. This information is local to the Steam client.
Request execution	The game's local purchasing server initiates a request through WebAPI on behalf of the Steam client using the *ISteamMicroTxn/InitTxn* call, including: 1. Order ID - 64-bit unique identifier for the purchase, 2. App ID - Unique identifier for the game whose asset is being traded, 3. ISO 631-9 code for the Steam client language, 4. ISO 4217 currency code for the client, substituted with a market-value currency rate if not present, 5. A list of items being traded, containing the 32-bit item ID, quantity, amount, description, and categories of each item.
Authorisation	If successful, the request attempt automatically notifies the Steam client to authorise the transaction along with transaction details and funding options. After user involvement, the client receives a notification regarding request

	result using the *ISteamUser::MicroTxnAuthorisationResponse_t* handler.
Result and completion	The purchasing server finalises the transaction through the *ISteamMicroTxn/FinalizeTxn* command with the order ID and app ID obtained earlier. A response is provided directly to the client regarding transaction completion or errors. This information is sent from the billing servers to the client.

Figure 5. Steps taken for Steam asset transaction (SteamWorks Foundation)

These stages are implemented during every micro-transaction or inter-player transaction on Steam. Some steps may be redundant when making multiple requests and packaging limits imposed by the API can cause processing delays.

III.3. Use in asset trading systems

Client-server models can utilise AJAX or HTTP protocols to enable transactions between players. In most cases for simplicity HTTPS protocols are used. Polling is enabled between clients and servers for inter-player transactions. This is crucial to maintain a consistent connection between the two stakeholders in the transaction.

The primary limitation of the client-server model is the possibility of server infection. Using the game purchasing server as a middleman for these assets also puts both the purchasing and billing tiers at risk of Denial of Service attacks. During a server-side overload when multiple transactions are being handled, the risk of DOS attacks becomes much greater. SteamWorksAPI acts as a workaround, facilitating the transaction. Moreover, the API implementation chances of transaction cancellation and information abandonment, reducing the risk of phishing. Forcing the usage of order IDs helps track any faulty transactions, but the client country and identifiers are only optional, reducing the possibility of tracing down any criminal activity.

III.4. Analysis with respect to criteria

III.4.1. Speed and reliability

Request speed and protocol overhead

Traditional client-server architectures afford higher speeds for data transfers than blockchains. Server layers are designed to offload different tasks to different groups, thereby reducing overload on a single server. Because clients directly interact with servers the operations pass through fewer intermediaries and are faster. The existing system scores a 3 in this criterion.

Device-side processing

For asset transactions, the load of the transaction processing is offloaded almost entirely to the server. The client is required to remain connected to the servers for the duration of the transaction. This is required until the transaction is authorised. There is little processing done on the client-side for packaging transaction requests. This criterion scores a 3.

III.4.2. Storage

Integrity of transaction data

Client-server transactions operate on the principle of atomicity. A transaction is either completed or not completed at all. Errors are handled and transaction data is kept isolated from other transactions, through special identifiers. However, when in transmission the transaction is at risk of waiting due to possible server overloads, mitigated by the P2P architecture. During transmission, data integrity can be compromised. 32-bit and 64-bit private and public keys serve to maintain transaction data integrity but can be easily broken during a brute-force attack. If used, the keys of a higher standard should be applied for maintaining security. This criterion scores a 2.

Integrity of asset data

Asset data is not transmitted during the transaction process. The asset data remains on the database layer of the server architecture used by Steam. Bypassing the front-end server from a client would be difficult due to a lack of direct interaction between the client and the database. Thanks to layer security, this criterion scores a 3.

Data loss prevention and tracing

Due to atomicity, transaction data is either entirely lost or verified when a transaction takes place. Polling is not always allowed for transaction purposes, losing all data related to the transaction. Repetitive occurrences may result in the loss of the asset from the client's inventory without their consent. In case of malicious transactions, the Steam IDs of both stakeholders are kept to better track and apprehend criminal activity. A blocklist is also implemented. This criterion scores a 1 due to inadequate rollback systems for data loss.

III.4.3. Computational power

Client-side processing and requirements

Steam operates using a thin client system. Almost all computational load is hence handled by the server. Zoning systems for servers also reduce the need for client computation for detecting the fastest server. In this regard, the criterion scores a 3.

Server-side processing and requirements

Due to thin clients, servers handle almost all computational workloads. The number of concurrent loads that servers can handle depends on their computational ability. Much computational power is wasted on the server-side if the client disconnects. In many instances, the servers can go offline for maintenance or overloading. A peer-to-peer architecture would mitigate this flaw. This criterion scores a 2 because servers can be upgraded to match client requirements.

Network connectivity and reliability

Consistent network connectivity is required for transaction completion. Lack of polling systems may cause transaction failures if the client network is not consistent. While maintaining security, it also increases the number of times a transaction may occur. This may lead to double spending if billing servers are still processing requests. Hence, this criterion scores a 1.

IV. Blockchain models

IV.1. Topology design

A blockchain operates on the peer-to-peer model, wherein every node on the network must be an independent client and server. It represents a distributed ledger system that is immutable, or not prone to data loss, and decentralised. There are three primary topologies which a blockchain network can use: private, public, and consortium networks.

IV.1.1. Public blockchain

A public blockchain is also called a permissionless blockchain. Any node can join the blockchain network and take part invalidation. However, to prevent malicious activity, proof of work algorithm (PoW) limits the rate of addition of blocks to the network, done through SHA-256 hash generation and NONCE searches.

This system of blockchain is typically used for cryptocurrency exchange and data/crypto mining. All steps in this topology are incentivised through cryptocurrency compensation. Node consensus and independence ensure data security and transparency.

IV.1.2. Private blockchain

A private blockchain works based on only authorised individuals taking part in transactions. It differs from a public blockchain with individual control of the blockchain, centralising its authority. That individual provides all read and write permissions and on-demand authorisation for operations. In most cases, a firewall protects access to the blockchain. This topology distributes the load between all nodes better and is more efficient. Firewalls reduce the need for proof-of-work security. Instead, proof of stake (PoS) is used, wherein a node's existing stake in transactions is the metric for selecting the next node to handle transaction approval.

Because it relies on existing data and does not vary based on computational ability, it is less resource-intensive and allows relatively thin clients to participate. Byzantine fault tolerance (BFT) uses a consensus of ⅔ of nodes present to approve a transaction. Because this topology is more restricted in access, it is used more in backend systems than front-end public implementations.

IV.1.3. Consortium blockchain

A consortium blockchain is also known as a public-permission blockchain. In this, a single authority/group of authorities controls write access to the blockchain, and public access is limited to reading and accessing content present on the nodes, and not taking part in transactions. This implementation of a blockchain operates on PoS and BFT because of the limited nodes participating in transactions, not needing more rigorous algorithms to retain security. Companies use this to allow consistently-updated company-specific data like financial data to be accessible in the public domain, to increase consumer trust and avoid litigation for non-transparency.

IV.2. Anatomy of a blockchain transaction

Each full node in a blockchain contains a copy of all transactional data, ensuring the verity of any data, and mitigating malicious activity. Different algorithms like Proof of Work (PoW) and Proof of Stake (PoS) are used by block miners/signers to validate a block. Based on these two algorithm categories, the system requirements of a blockchain vary. A transaction can be broken down into the creation and digital signature of the transaction, propagation (flooding) of the transaction, creation of the block, and the addition and verification of the block to the blockchain.

Stage	Description
Creation and digital signature	Transactions are created through a data structure representing a transfer of value, with supplementary data like the logic of transfer proceedings and source and destination addresses. All data is packaged into a single unit secured using a private key. This unit is known as a digital signature.
Propagation of transaction	The digital signature is transmitted to multiple peer nodes for validation of the transaction. A consensus of peer nodes is used to validate the transaction.
Creation of block	If validated, all digital signatures associated with the transaction are put into a single data block. By the block creation stage, the transaction is effectively considered

	confirmed.
Addition to ledger and block verification	When the block is added to the ledger, it obtains the SHA hash of the block behind it and a nonce to be solved. The hash pointer to the block ahead of it is calculated with the nonce using either the Proof of Work (PoW) or Proof of Stake (PoS) algorithm depending on the topography of blockchain. 1. Proof of Work (PoW) - This method first verifies the block data before nonce calculation. A hash is generated using a random value. If this hash does not meet the criteria imposed by the block consensus, the hash is again generated randomly using the nonce, until the correct hash is found. It relies on a brute-force process to obtain the correct hash and is computationally taxing because multiple miners are involved in solving the nonce. In compensation, the miner obtains a given amount of new currency and its crypto wallet is updated to match. 2. Proof of Stake (PoS) - This method verifies the block data before nonce-based calculation. Instead of solving the nonce to generate a block hash, a node is randomly selected through its investment in the blockchain i.e. the number of coins it possesses and their age. This is referred to as the stake. A hash is generated using these parameters along with the nonce.

	If the hash does not meet the consensus, it is again generated with a new timestamp until the consensus is met. The selected node is compensated with newly minted currency, and its crypto wallet is updated to match. Using the generated hash, the block adds to the ledger, with the previous and next blocks matching their hash pointers with the new block to verify its addition.

Figure 6. Steps for transaction done through a blockchain (Medium)

These steps are performed every time a block is added to the system or a transaction takes place. All blocks in the ledger undergo verification when new blocks are added, and typically six verifications are required to finalise the status of a block on the network. Peer confirmation and multiple verification attempts ensure the security of all blocks.

In larger implementations, though the time duration may be equal, the difference in computational power required for each verification method because of simultaneous computation in PoW. This will increase as the blockchain scales up. In this sense, the PoS algorithm is more efficient and sustainable.

IV.2. Use in asset trading systems

A large-scale asset trading system would use a PoS blockchain, ensuring most users can participate in transactions. A consortium blockchain set is ideal for usage in asset management and trading systems. It ensures individual asset security during trades by keeping one blockchain for asset storage and another PoS blockchain for asset transaction.

IV.3. Analysis with respect to criteria

IV.3.1. Speed and reliability

Request speed and protocol overhead

All decentralised systems require more time to process requests than centralised client-server architectures. The time required to broadcast and validate a transaction across a blockchain network causes this. It causes higher latency and further slows down the request speed. For PoW, a limit is kept on the maximum transaction rate in the blockchain. Thus, proof-of-work would further increase the latency for access. Algorithms prioritising closer nodes and PoS mitigate this. However, it remains prevalent in blockchain transactions.

Because this latency does not severely affect the working of the model, some redundancies can be overlooked. This criterion has been given a score of 2 as a result.

Device-side processing

Any device which accesses a blockchain and the data it contains must partake in a blockchain transaction. The device thus takes both the computational load of accessing and pulling data from the other nodes and the computational load of partaking in a transaction through nonce calculation and mining/forging. This load is typically higher than that of a client device accessing a server. In an asset trading application, it will not affect the ability of the device to partake in asset trading, while increasing the security of the asset trade, aside from the latency. As a result, this criterion scores a 3.

IV.3.2. Storage

Integrity of transaction data

Because a blockchain operates through a peer-to-peer network, all transactions are validated by a majority of operating nodes, using a Byzantine fault tolerance system on the network. Thanks to this, the validity of the transaction is kept intact. However, if a single authority owns/operates 51% of more of the nodes that control access to the blockchain, the transaction data can tamper. However, considering a network that operates with over 21 million concurrent users (Statista) this event is extremely unfeasible and unlikely. Hence, this criterion scores a 3.

Integrity of asset data

Rules similar to those specified for transaction data would apply to the asset data, as a consensus is required for any transaction involving the blockchain. Considering the scale of such a blockchain, the assets would remain as safe, if not safer than the transaction data, direct interaction does not occur between the user and asset. Hence, this criterion scores a 3.

Data loss prevention and tracing

A blockchain functions by allocating hashes to individual blocks, along with the previous and next blocks' hashes connected cryptographically. The block itself contains transaction data including the addresses of those involved. Hence, to trace an individual transaction, the TxHash (transaction hash) value is required. Tampering with a single block involves tampering with all blocks simultaneously due to linking of hashes. This would be completely unfeasible at a larger scale. Continuous node verification further reduces the likelihood of this happening. This criterion, therefore, scores a 3.

IV.3.3. Computational power

Client-side processing and requirements

The client-side processing requirements differ for different implementations of a blockchain. For Ethereum, the system requirements for operating a node in the least resource-intensive form are as follows:

Component	Minimum requirement (Ethereum.org)	Recommended requirements (Ethereum.org)
CPU	dual-core or higher	quad-core or higher
RAM	4 GB RAM if using SSD, 8 GB+ RAM if using HDD	16 GB+ RAM
Storage	N/A	500+ GB SSD
Network speed	8 Mbit/s	25+ Mbit/s

Figure 7. System requirements for Ethereum platform (Ethereum Foundation)

As seen, the minimum system requirements for operating a full node are more input/output intensive than processing intensive. This is because the node merely stays in sync with the blockchain (Ethereum.org). More powerful processing units are beneficial to operation.
Light nodes that can check the validity of transactions and store only the header node, requesting everything else, also exist. This is beneficial for lower-powered implementations or mobile

devices. A PoS-based Ethereum update is being developed, which should reduce requirements further.

However, the system requirements for the Steam client are lower than those mentioned above. That said, on Steam statistics most clients operate on quad-core processors and high-capacity storage devices which fulfil recommended node specifications (Valve Corporation). Existing systems running below these requirements can operate as light nodes, and still be able to conduct inter-player transactions. This criterion scores a 2.

Server-side processing and requirements

The server-side and client-side processing requirements are identical because of the peer-to-peer model. More powerful servers, if running virtual machines, can operate multiple nodes at a time thanks to this. This criterion thus scores a 3.

Network connectivity and reliability

Blockchain computation is very network intensive. All operations are synced with other nodes. For larger node implementations, blockchains may require several hours to fully sync. This is common to both PoW and PoS systems. Nodes may not be active at all times, requiring syncs when reconnecting.

Ethereum provides several options for syncs, varying in storage requirements and time. This provides flexibility and does not require constant uninterrupted internet connection using long-polling requests, but still relies heavily on bandwidth availability. The Steam client, by default, requires internet access and can consume a large amount of bandwidth at times for server operations (offloaded onto the blockchain). Thus, this criterion scores a 2.

V. Comparison

Aspect	Client-server	Blockchain
SPEED AND RELIABILITY		
Request speed and protocol overhead	3	2
Device-side processing	3	3
STORAGE		
Integrity of transaction data	2	3
Integrity of asset data	3	3
Data loss prevention and tracking	1	3
COMPUTATIONAL POWER		
Client-side processing and requirements	3	2
Server-side processing and requirements	2	3
Network connectivity and reliability	1	2
TOTAL	18	21

Figure 8. Table of comparison of criteria.

V.1. Speed and reliability

Client-server models have an advantage over blockchain models as they are simpler to implement. Although blockchains can be programmed easily through a variety of languages, the speed of a peer-to-peer architecture lacks in comparison to a client-server architecture due to direct connections between servers and clients. The difference in speed is not negligible but is not a major drawback to this model.

V.2. Storage

Asset data and transaction data are significantly more secure in a blockchain network than a client-server model. This is the most significant improvement that blockchains can provide for transactions. Additionally, the traceability of transactions is built into the blockchain system due to complete transparency. Client-server models can be modified to provide more security, but blockchains provide significant improvements in this criterion.

V.3. Computational power

Both blockchains and client-server architectures rely on heavy computation. In a blockchain, this is distributed more between nodes, because of peer-to-peer architectures. A higher load on the client limits the prospective users who can use the system in its full capacity. Light nodes exist but these may not provide a full computation of transactions. However, this distribution of computation also means that those with more powerful computers can also share their resources more, and balances workloads on a larger scale. The reliance on network connectivity is high for both models, but blockchains which use polling rely less on networks for transaction verification. A client-server model rolls back transactions in the case of network errors. Blockchains thus provide more consistency here.

VI. Conclusion

As a whole, based on this investigation, one can have a clear position on the feasibility of a blockchain system in the management of assets in games involving inter-player asset exchange. This in-depth analysis of both models allows us to understand the models' contextual relation and their individual strengths and weaknesses.

The client-server model is very well supported and is currently the de-facto option for this purpose. It is widely implemented but lacks security and consistency, due to which blockchains are evaluated as a substitute. Blockchains have an advantage of transparency and redundancy, reducing server costs in case of node failure.

Blockchain also features a more distributed computational workload, allowing for less powerful servers and thus less consumption of resources, a major issue with both models. Blockchain has future scope in further optimising transactions through proof-of-stake implementations.
The main limiting factor for blockchains is the computation required for existing proof-of-work algorithms and storage requirements for nodes. Request speed tends to be lower and client devices have higher computational loads. It is also a recent concept and cannot be widely used until proof-of-stake policies are further tested. As more people begin to use blockchains, the model will become more and more optimised.

Today, one can conclude that blockchains should be implemented in beta variants for asset management systems for larger applications. Small implementations exist that are growing in popularity, but the model needs to be more refined to become widespread.

VI.0.1. Evaluation

This evaluation only looks at the theoretical capabilities of each system based on existing data overheads. Several factors outside the scope of my essay may play a role in the correct model to choose. Furthermore, a very academic viewpoint was used for this evaluation, and the theoretical differences may not be as perceptible for request times and other workloads. In very large applications, the future scope of the models used is important for system functioning and viability.

VII. Bibliography

Clement, J. “Number of Steam Users 2020 | Statista.” *Statista*, Feb. 2021, https://www.statista.com/statistics/308330/number-stream-users/.

Ethereum Foundation. “Ethereum Development Documentation | Ethereum.Org.” *Ethereum*, edited by @wackerow, 19 Jan. 2021, https://ethereum.org/en/developers/docs.

Linux Foundation. *Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus*. Edited by Hyperledger, vol. 1, 2018, https://www.hyperledger.org/wp-content/uploads/2017/08/HyperLedger_Arch_WG_Paper_1_Consensus.pdf. Hyperledger Documentation.

Medium. “CoinBundle – Medium.” *Medium*, edited by Editors at Medium, Medium, 2018, https://medium.com/coinbundle.

Valve Corporation Inc. “Steam Hardware & Software Survey.” *Steam*, edited by Steam, Valve Corporation, Jan. 2021, https://store.steampowered.com/hwsurvey.

Weber, Ingo, et al. “On Availability for Blockchain-Based Systems.” *IEEE Explore*, vol., no., Oct. 2017, pp. 64–73, doi:10.1109/SRDS.2017.15. IEEE.

Baughman, N.E., and B.N. Levine. "Cheat-Proof Playout For Centralized And Distributed Online Games". *Proceedings IEEE INFOCOM 2001. Conference On Computer Communications. Twentieth Annual Joint Conference Of The IEEE Computer And Communications Society (Cat. No.01CH37213)*, pp. 8-10. *IEEE*, doi:10.1109/infcom.2001.916692. Accessed 23 Jan 2021.

GauthierDickey, Chris et al. "Low Latency And Cheat-Proof Event Ordering For Peer-To-Peer Games". *Proceedings Of The 14Th International Workshop On Network And Operating Systems Support For Digital Audio And Video - NOSSDAV '04*, 2004, pp. 4-5. *ACM Press*, doi:10.1145/1005847.1005877. Accessed 22 Jan 2021.

Knutsson, B. et al. "Peer-To-Peer Support For Massively Multiplayer Games". *IEEE INFOCOM 2004*, pp. 1-3. *IEEE*, doi:10.1109/infcom.2004.1354485. Accessed 23 Jan 2021.

Schiele, Gregor et al. "Requirements Of Peer-To-Peer-Based Massively Multiplayer Online Gaming". *Seventh IEEE International Symposium On Cluster Computing And The Grid (Ccgrid '07)*, 2007, pp. 1-5. *IEEE*, doi:10.1109/ccgrid.2007.97. Accessed 21 Jan 2021.

Shun Yao, and Jess Port Telles. *Method And System For Facilitating Online Gaming*. No. 2015/0089595, 2015.

Shun-Yun Hu et al. "VON: A Scalable Peer-To-Peer Network For Virtual Environments". *IEEE Network*, vol 20, no. 4, 2006, pp. 22-31. *Institute Of Electrical And Electronics Engineers (IEEE)*, doi:10.1109/mnet.2006.1668400. Accessed 21 Jan 2021.

Yu, Anthony (Peiqun), and Son T. Vuong. "MOPAR". *Proceedings Of The International Workshop On Network And Operating Systems Support For Digital Audio And Video - NOSSDAV '05*, 2005, pp. 101-103. *ACM Press*, doi:10.1145/1065983.1066007. Accessed 21 Jan 2021.

Dimitriou, Kostas, and Markos Hatzitaskos. Core Computer Science For the IB Diploma Program (International Baccalaureate). Express Publishing, 2015.

Statista. Number of peak concurrent Steam users from January 2013 to December 2020. *Statista.* February 2021. https://www.statista.com/statistics/308330/number-stream-users/. Accessed 2 February 2021.

Valve Corporation. Steam Hardware and Software Survey. January 2021. *Steam.* https://store.steampowered.com/hwsurvey. 2 February 2021.