GÜFBED/*GUSTIJ* (2020) 10 (1): 163-173 **Gümüşhane Üniversitesi Fen Bilimleri Enstitüsü Dergisi**
DOI: 10.17714/gumusfenbil.518689 Araştırma Makalesi / *Research Article*

Evaluating a Player's Network Class in a Multiplayer Game with Fuzzy Logic

Video Oyunlarında Bulanık Mantık ile Ağ Sınıfı Değerlendirilmesi

Tunç UZLU[a], Ediz ŞAYKOL*[b]

Beykent University, Department of Computer Engineering, Ayazaga, 34396, Istanbul, Turkey

• Geliş tarihi / *Received*: 28.01.2019 • Düzeltilerek geliş tarihi / *Received in revised form*: 28.08.2019 • Kabul tarihi / *Accepted*: 05.11.2019

Abstract

In a multiplayer game environment, smoothness of a game depends on factors such as game's netcode, player's hardware, network connection and server's response time. Players with bad network conditions and spiking network synchronization is always a problem for multiplayer games for both PCs and gaming devices based interactive sessions. Previous implementations to prevent such occasions involve disconnection based on their connection statistics. However usually used schema involves constant boundary values and mostly biased for disconnection decision. Disconnecting players with medium to worse network status is typically not an optimal practice as rendering the game unplayable for the person as if we think about player's hardly earned money. Today, fast CPU and graphics processors offer rendering of game and receiving the game state with high frame rates. Hence, quality of the game depends on network connection of participants. Games prefer measuring latency between the server and the player by sending simple ping packets similar to well-known Unix tool. An administrator evaluates this value whether the player should be disconnected or not. In this paper, we discuss evaluating the player's complex network statistics and automatically deciding for absence of the player with pattern recognition techniques.

Keywords: Fuzzy Logic, Gaming, Latency Analysis, Pattern Recognition

Öz

Çok oyunculu bir oyun ortamında, bir oyunun verimliliği; oyunun eşleme kodu, ağ bağlantısı ve oyun kurucunun yanıt süresi gibi çeşitli faktörlere bağlıdır. Kötü ağ koşulu, bağlantıların kesilmesi ve eşlemenin bozulması çok oyunculu oyunlarda büyük bir sorun oluşturur. Bunu önlemek için önceki uygulamalar, bağlantı istatistiklerini temel alarak oyuncuları ayırmayı ve sınıflandırmayı içerir. Bununla birlikte, genellikle kullanılan ticari yöntem, sabit sınır değerlerinden faydalanır ve çoğunlukla bağlantı kesme kararı uygular. Oyuncuların orta ve kötü durumdaki ağ sınıfında bulunmaları ve oyun dışı bırakılmaları finansal açıdan verimli bir uygulama değildir. Günümüzde hızlı CPU ve grafik işlemciler, oyunların tatminkâr seviyede hesaplanmasını sağlamakta ve oyunun verimi katılımcıların ağ bağlantısına bağlı kalmaktadır. Genellikle oyunlar yaygın bilinen Unix aracı olan ping'e benzer paketler göndererek sunucu ile istemci oyuncu arasındaki gecikmeyi ölçmeyi tercih eder. Ayrıca oyun dışı bırakılma kararı bir yönetici tarafından değerlendirilir. Bu makalede, oyun dışı bırakma kararını ağ istatistikleri ve örüntü tanıma tekniklerinden faydalanarak otomatik olarak değerlendirmeyi ele alıyoruz.

Anahtar kelimeler: *Bulanık Mantık, Video Oyunu, Gecikme Analizi, Örüntü Tanıma*

*[b] Ediz ŞAYKOL; ediz.saykol@beykent.edu.tr, Tel: (0212) 444 19 97, orcid.org/0000-0002-8950-5114
[a] orcid.org/0000-0001-9366-6677

1. Introduction

In this study, evaluating the player's complex network statistics and classifying players, based on corresponding input data is discussed. Creating a game world that is consistent across all internet players depends on one challenge; latency in network transport. Interactiveness and fairness of these virtual environments must be maintained via quick distribution of game state across participants and the centralized computing unit of the game provider. While factors that affect quality of service varies a lot, transmission problems with one player's network must be individual and should not take whole topology down. Sensitivity of network delays are relational to the genre of the game and implementation. Exploiting bandwidth and computation power of player nodes may be possible with peer-to-peer architectures and live migration techniques of central grid (Chan et al., 2007), but architectural changes does not eliminate the need for latency analysis.

Fuzzy logic is highly involved for various reasons. First, we need a smoother way to classify players by their connection statistics, because we have conditions such as very low connection speed, medium latency of transfer of packages, high-to-very-high player concentration, etc. Disconnecting players with latency greater than and equal to a fixed level may be the easiest solution. However specific game sessions, genre of the game, number of other opponents, absence of any background updating process and may other factors all involved in determination of quality factor of the player's connection capacity. Another important benefit of using fuzzy logic is to detect players that prone to disconnect. This is important for peer to peer sessions where one player is considered as a host. Other players connect to this host player and act as client players. Selecting a host player is very important from the perspective of game synchronization latency, and this host player should be a middle-man (balancing the network for all players) (Chen et al., 2005).

Broadband network speed has increased dramatically since Arpanet. However online gaming performance is not about the network bandwidth. Even the throughput is not enough to detect problematic networks on online gaming. The reason for delay in multiplayer games is a consequence of various reasons including, but not limited to, geographical location (propagation delay), wide area network access type, transient conditions (congestion in network during transport) or improperly designed Quality of Service (QOS) mechanism, or lack of the QOS all together (Zander et al., 2005).

While measurement of the ping values is the de-facto method for detecting latency between two nodes (URL-2, 2005). Threshold intervals have been used with ping values to determine players with bad connection. However, some problems within the server (in server-to-client sessions) may affect latency value between the nodes. These server issues may include denial of service attacks, high CPU usage and shared memory. Attacks from outbound network may be slowed down with modern network hardware such as firewall or load-balancer.

It is harder to utilize such technologies with gaming hosts as architectures vary significantly. CPU usage is also dramatically increased when the number of players is high. Sometimes other process or kernel tasks may utilize CPU in high ratios; such as updating scenarios. Shared resources with virtual servers is another problem. Over-quota usage of other virtual clusters or bad disk swap spaces may slow down memory operations. Some players may have nonoptimal client configurations such as bad player sync timer, update rate or network interpolation values (Henderson, 2001). When combined with lag compensation, it shows us, not only the server, but also other players in the session may also cause spikes in latency. This concludes that constant thresholds are not optimal for the purpose.

To elaborate on the importance of this problem, a gaming operator in Taiwan dedicates more than 4000Mbps link connectivity purely for their gaming infrastructure (Chen et al., 2005). This number is very high when compared to the total bandwidth of even some countries; and hence any improvement in this online gaming environment would be valuable. Main factors like latency, jitter, packet loss (we study these elements as main inputs to classification) and auxiliary factors like downstream speed, borderline SNR (we also study these in our fuzzy evaluation to down-class players that are prone to disconnect because of bad network conditions) are some basic concepts to analyze. Geographical locations of players, genre of the game and protocols are also decisive factors in this manner. Connections passing through country outbound known suffer from high latencies. Playing in an isolated environment in a country (or even city) is preferable.

Another example game utilizes a periodic pulse to detect anomalies with latency measurements within players (URL-2, 2005) and this consistency may fit very well to a specific genre. However, when some other designs concerned (for example; applications that does not involve client-server architecture or completely different type of game, like map or turn based games) may not fit well with constant pulse frequency. The server may be transmitting events with a periodic and constant time slice, similar to round-robin method. Within "map" based games (games that mimics huge chessboards), the server's synchronization is generally more periodic and the client's actions are in bursts, successive and much more predictable. This paradigm actually results in some kind of spatial locality (Chen et al., 2005) affecting periodic latency controls over time.

Based on these observations and analysis, we present a fuzzy logic scheme for evaluating the player's complex network statistics and classifying players in online video games. We start with explaining networking concepts that affect online gaming experience in Section 2. Then in Section 3, we continue with evaluating results with Naïve-Bayes classifier usage in online gaming with sample real data. Section 4 and 5 provide explanations with fuzzy logic and learning schemes. Finally, we discuss the results in Section 6 and conclude paper.

2. Related Networking Concepts

Bandwidth, latency, jitter and packet loss are the most important network terminology when gaming is involved. Facts about the wireless networks worth mentioning. Lost and discarded packets cause incomplete processing of game state. With most game types, delayed packets are discarded when they are not received in a time frame. To create reliable and fair game state, packets must be transmitted with minimal delay (latency effect), closed to zero loss (packet loss effect); and there should be enough bandwidth available (dependable QOS on transport network). To classify players correctly, latency measurements should not yield spiking results; thus, should be consistent even if there is high latency (jitter effect). As wireless is designed to work half-duplex, has unique issues in term of signal penetration, congestion and lost packets (Reid and Seide, 2002). Transport characteristics of a wireless connection are different compared to direct-line connection; evaluation of wireless connection statistics is intended to be included in further study.

Bandwidth is the capacity of data receiving and sending in a time frame, here per seconds. Latency is the major factor in order to determine quality of a network. It is mainly used as the only element to detect a game player's network status, by most popular games. Ping is how fast a response is received after sending a request. Most interactive games need response times between 100-1000 milliseconds depending on the game genre (Harcsik et al., 2007). Therefore, faster the Ping round-trip is, lower the latency. It is an ICMP protocol service, measured in milliseconds, and generally contains a small payload data.

Packet loss is percentage of packets lost with respect to packets sent. The reasons include network congestion, faulty router configurations, bad lines or utilizing unreliable by nature network access, such as GSM, for multiplayer gaming.

Jitter is the variance in ping capacity that measured in milliseconds. Zero jitter means ping results are stable, drawing a straight line. Examples may be well-adapted lag compensation and smooth VOIP sessions (e.g. variable bitrate codecs). When a network traffic occurs, there are delays of transmission. This random variation is called a delay jitter. In an online gaming environment, these delays are critical and unpredictable. In video communications, poses are received one after another and displayed in a fixed time frame. Any delay in transmission will result with paused frames. Besides high jitter may also cause packet loss. Because of rapid receival of delayed packets, operating system or networking subsystem may not be able to process the input (e.g. receiver buffer may be full and may overflow, resulting a race condition) (Joe, 1996). In gaming jitter is equally and even more important. Packet payloads are much smaller compared to real time video transmission and jitter is important bidirectionally because input from the client side is also transferred through network.

As shown in (Henderson, 2001) that measuring application-level latency is preferable compared to network-level latency (by utilizing ICMP packets). The reason for this is sending additional data may increase the network traffic going through the server and alter the behavior of clients.

Line statistics are measurements of wide area network connection of the peer. Although they have no relation to connection speed, they provide valuable information for detecting players that are

prone to drop out. Statistical information of internet service provider connection cannot be obtained by using standard home networking methods. In order to retrieve this information, a direct access to player's router is necessary. This makes players from enterprise and public spaces flagged right away. Most cable connections expose downstream power, upstream power and signal to noise ratio. Downstream and upstream power values that are closed to boundary values are likely to cause reconnections randomly. Values that are out of defined values are certainly going to cause drop outs. They are valuable information to detect players that are prone to disconnects.

Although here only a few values are used and described, there are additional information that cannot be fetched from home networking equipment. Signal to noise ratio is a valuable resource to estimate packet loss without running a complex loss analysis. Combined with calculated latency, it provides a deep insight into the remote peer connection quality. Digital subscriber line connections also expose signal to noise ratio, the logic is the same as cable connections. They also expose line attenuation, which is the counterpart of downstream power.

Thinking about a star topology, favoring few players and ignoring most other players is going to affect the quality of game session badly in the case of disconnecting the host player. Then, the host has to be transferred to another player. There are various problems with this scenario albeit most modern games implement such occasions very well. Selection of a new host takes time. Especially if the new host player is selected using a heuristic method or by an artificial intelligence algorithm, like our fuzzy evaluation method here. As measurement of network statistics and calculating the result is going to take time, even minutes, this is going to create plenty of inconvenience, especially with games with short sessions (e.g. 10 minutes). All players are going to lose their focus, maybe some of them are going to leave the room or ragefully quit from the game. Even if we manage to select a very good new host player, the game may not be able to continue because of desynchronization, corruption of the state, timeouts or loss of connection between the central servers. As the game is still running at the back, the central processing unit and the memory is completely dedicated to the game session. Determination of the host player is easy from the main game menu, because menus or pre-game events demand much less hardware resources compared to real game session. Being in an active game session limits the hardware resources for our fuzzy logic algorithm, especially when gaming consoles are considered. Games made for consoles usually tops hardware utilization and leaving spare memory space for only kernel plugins of the operating system. By considering all these reasons, our fuzzy logic design must classify players that prone to disconnect. By leaving out this group of users, one is able to decrease the probability of migration of the host to another player dramatically.

3. Naïve-Bayes Classifier Method

Here, our aim is to show that Naïve-bayes type classification scheme could be useful in online gaming to detect the players that should be deported. Here, the features that we use are download speed (Mb/s), upload speed (Mb/s), ping (ms), jitter (ms), packet loss (%), is-deported and location.

Sample input set, gathered from real users, consists 25 entries with 14 acceptable and 11 unacceptable entries. 10 of them are happy while 15 of them complains about their internet connectivity. Latency requirements depend on netcode and level of interactivity.

Constructing the lookup table by looking at Chariot reference values (URL-1, 2004) and interviewing system administrators. There are two ranges for jitter and packet loss and 3 ranges for the ping. Values defined by Cisco are <10 ms for jitter, <50 ms for delay and 0.5% for packet loss.

The classifier can be learned from example data shown in Table 2. Reference values for a typical session is shown in Table 1 and naive assumption using Bayes' theorem in simplified form (Shanahan, 2000) is shown in Equation 1.

Table 1. Connection reference classification

	Ping	Jitter	Packet loss
Low	<= 50	< 10	[0-1]
Med	(50-90]		
High	> 90	>= 10	>1

Table 2. Number of class values same as the example from the input set

	ping=H	jitter=L	pl=H
Yes (14)	6	3	4
No (11)	4	10	0

Following statistics belong to player; ping=95, jitter=10, package_loss=1.5 therefore classes; ping=H, jitter=L, package_loss=H.

$$P(Xj \mid Yi) = (se + c\,x\,t) / (st + s) \quad (1)$$

where
se → number of samples for X=Xj and Y=Yi
c → equivalent sample size constant
t → estimate of traits for P(Xj, Yi)
st → number of samples for Y=Yi
prior P values: P(yes)=0.56, P(no)=0.44
Example parameters for P(X=ping:H | Y=yes) should be; se=6, c=2, t=⅓ and st=14.
Here the c value, equivalent sample size, is constant and selected whimsically.
P0: P(ping=H, y); 0.42
P1: P(ping=H, n); 0.36
J0: P(jitter=L, y); 0.25
J1: P(jitter=L, n); 0.84
L0: P(pl=H, y); 0.31
L1: P(pl=H, n); 0.08
P(yes)*P0*J0*L0 is equal to value of 0.018 and P(no)*P1*J1*L1 equals to 0.011.
This results that sample player should be disconnected from the session.

4. Fuzzy Method

There are three input classes and one output class in our fuzzy method. As for the output, there is not a deportation Boolean class as in Naïve-Bayes classifier. The worst-condition class is used to determine these criteria. Descriptions of 4 different classes are shown in Table 3.

For downstream power, the gaussian membership function has been selected. It is the perfect match as smoothness is desired and conditions start from worst, worse, good to best and then goes down the hill symmetrically. Optimal connection has the steepest curve as it demands a little bit closer range to zero millivolts.

Table 3. Definitions of classes for resulting fuzzy evaluations

Class name	*Description*
Disconnect	Worst connection quality. Players are prone to be deported from the game.
Slow	Medium line quality. The game is playable, but performance is not optimal.
Fast	Like the optimal class, it is also expected to provide the best experience in this class. This is not the optimal class, but this is actually the targeted class for most players. Lagging, shuttering, texture popping are not expected in this class.
Optimal	The most optimal class. Every player wants to be in this class.

Bandwidth is important to some degree. The difference between 60 megabits per second and 90 megabits per second is negligible for our case. A triangular member function is employed to show the direct proportional relation. After the threshold value, the result is constant.

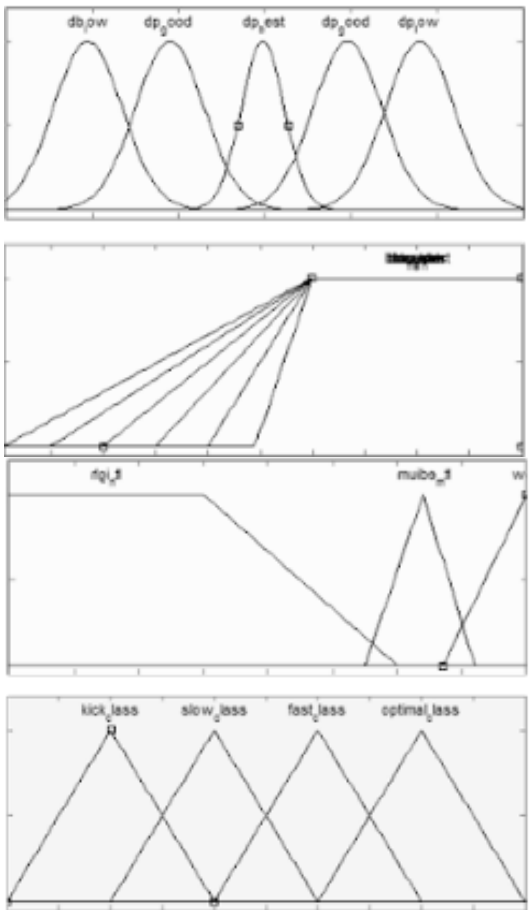

Figure 1. Membership functions for downstream power, bandwidth, latency and the output respectively.

Combination of triangular member functions and a sigmoidal (for lt_high) member functions are used here. Lower values indicate the low latency hence better the connection is. For high to very high latency, after the threshold value, the result is almost constant as the resulting game session is barely playable. In the graph, game's demanding latency is medium level so it can be modeled as a quadratic relation. A scatter plot is drawn from the dataset of 1000 values that we are going to use as input for fuzzy evaluations later. The correlation between latency and disconnection ratio is shown in Figure 1.

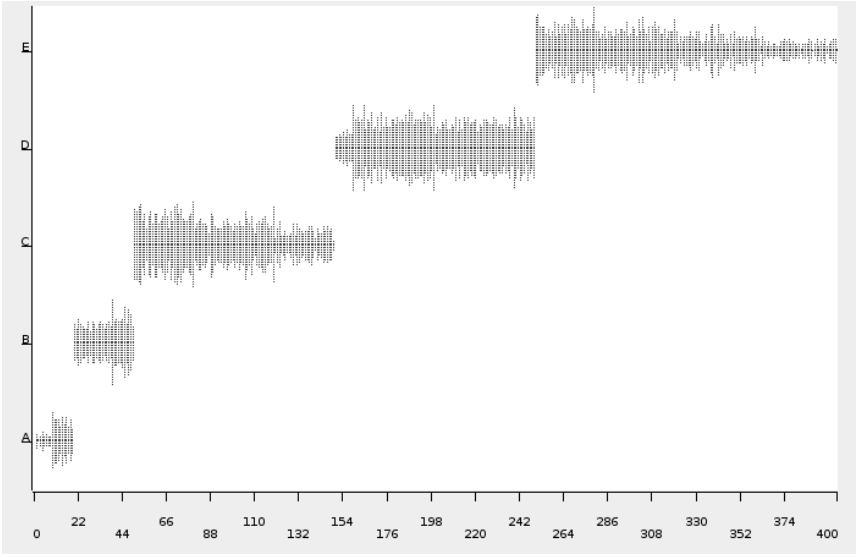

Figure 2. Scatter plot for predefined mathematical classification (classes A to E) and average latency.

Besides, if the game publishers share bounding values, every player can calculate their network quality index, or class, without the need of any programming or complex network knowledge. A simulation within Matlab has been drawn in Figure 3. Inputs of p=246, dp= -15.9, bw=30.2 are fed into the simulation and result of 0.242 (class II; slow) is obtained.

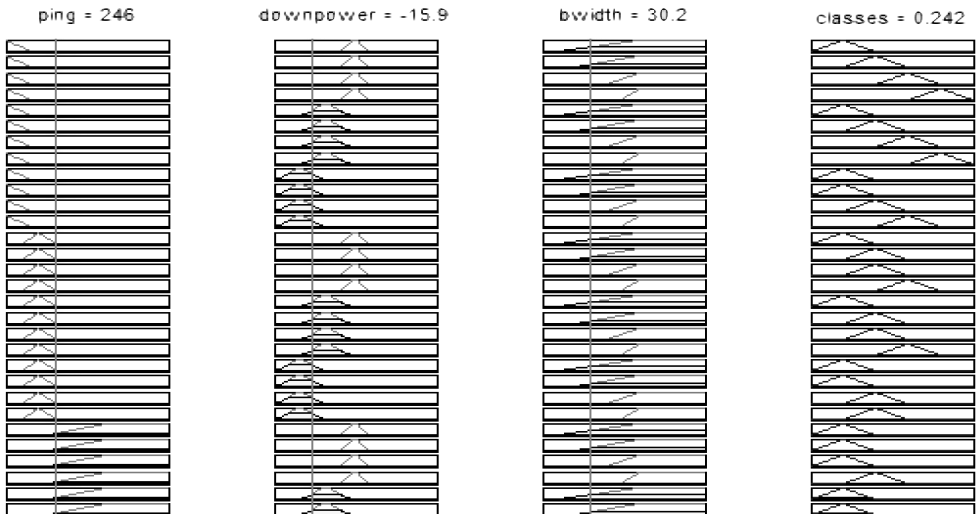

Figure 3. Application of rules of inputs in Matlab environment

Triangular member functions are selected for four distinct classes. Values in the range are equally divided and a there is no favor against any class. A scatter plot of classification and latency is drawn in Figure 2. In the graph, classification is made with cut-off values and entries that does not belong any class are taken out.

The rules are predefined for most common genres; however, it is imperative to tweak the values on specific games. The rules for in-game measurements and pre-game measurements may differ (therefore two different rule sets may be defined for re-evaluation). There are 3 rule sets while each set consists 12 distinct rules, forming a combination of 36 rules and set 3 is shown in Figure 4.

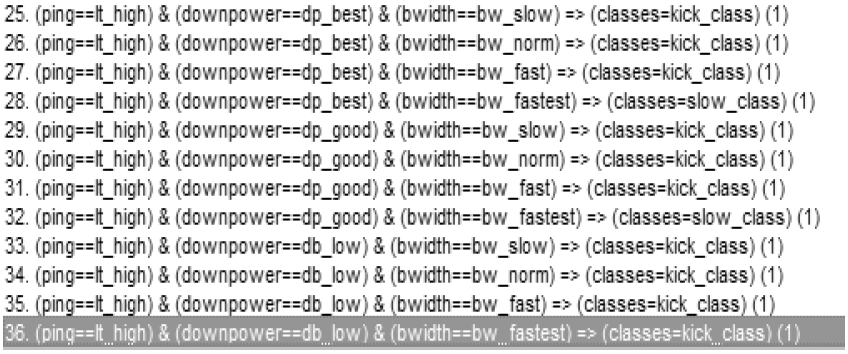

Figure 4. Sample fuzzy evaluation rules when *"Ping" input is "High"*.

5. Neural Network Learner

Our only output is a Boolean value indicating the deduction of player's presence. We try to estimate this result with latency, jitter and packet loss. Within the fuzzy logic method, we created a result set of 4 distinct classes while the members of the worst class were disconnected from the gaming session. Connection statistics were also a part of the classification, re-evaluating the borderline inputs. Hence members with very good connection, but unstable connection were never becoming a part of the best-ranked classes. With the neural network learner, instead of classification, we deduce that a disconnection should occur or not.

Game drop-outs (leaving from a game session because of a negative reason, technical or otherwise) because of latency or performance factors as proposed in (Lebres et al., 2018), when a multiple regression performed, obtained R^2 is 0.539 for five different predictors. These are latency/performance, features, community, support and fairness. It is shown that %53.9 of the drop-out variation is bringed forth these five predictors. Latency/performance, which is the important factor in our study, has a standardized beta of 0.346 that is the second largest after fairness (0.444). The workflow of data preparation parts is shown in Figure 5. Because of the complexity of processing and allowance of feedback within increasing learning process, utilization of distinct classes may not be a necessity. Our learner model uses back-propagation neural network learner (RProp learner) and fuzzy rule learner. The workflow of the learners as a continuation of the preparation part is shown in Figure 5.

A dataset of 3619 entries is obtained from a broadband discussion mail list from the USA. 10 sample rows are given in Table 4. Columns I1 to I5 of the dataset are download speed (Mb/s), upload speed (Mb/s), station-to-station latency (milliseconds), jitter (milliseconds) and packet loss (percentage) respectively. Values are grouped by mean opinion score. Download and upload speeds are unused, intended to use at future work. Rows are selected as two-tuples from five distinct groups varying from best to worse connection quality. Distribution of 3619 rows and their corresponding delay density is shown in Figure 6. The O1 column (output) is calculated manually and partially used in the learner (hence capitals for Boolean; Yes and No are overstriked). It is the output that we try to guess with Neural Learner. In the example table (Table 4), O1 column has been calculated manually with cut-off intervals related to I3 (latency), I4 (jitter) and I5 (packet loss) columns. Example rows in the Table 4 are carefully selected to show possible 5 different connection quality groups; starting from the best (green) to the worst (red) condition.

Back-propagation has three distinct layers. Before and calculation takes place, weights for the nodes are random (Rumelhart et al., 1986). Back-propagation is a supervised network and widely used for pattern classification and predictions (Mohanty et al., 2010). It is mainly suitable for regression, but it is shown in (Lee and Chang, 2016) that learner may be also utilized in concealment of packet loss occurrences.

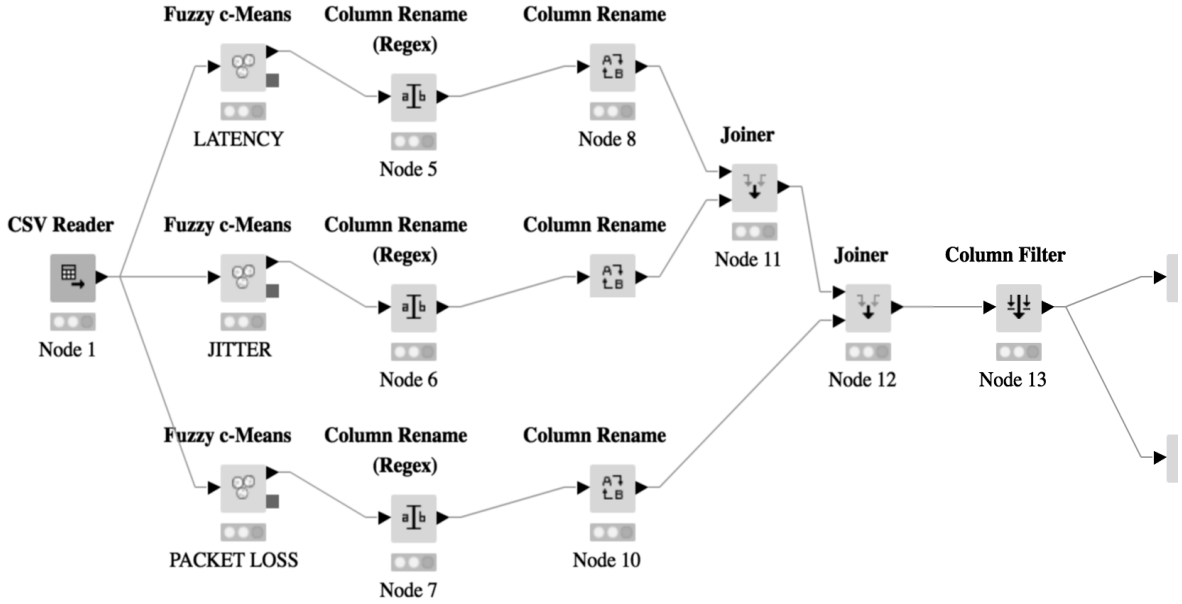

Figure 5.1. Knime workflow environment; preparation part.

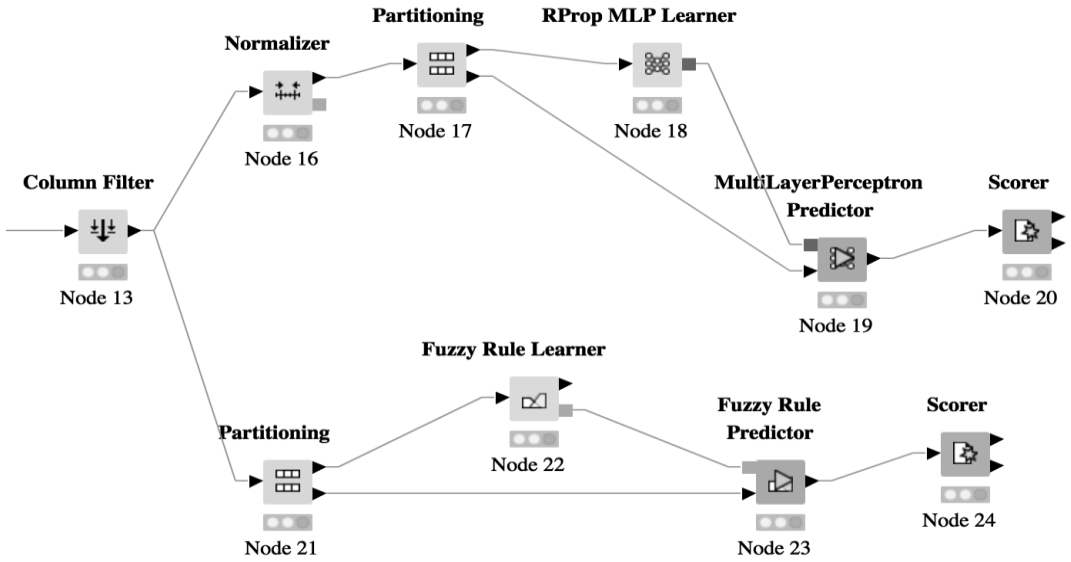

Figure 5. Knime workflow environment; scoring part.

Table 4. Cherry-picked 10 rows from the dataset

I1	I2	I3	I4	I5	O1	Location
31.78	3.67	29	2	0	₦	Seattle WA
5.90	0.44	21	0	0	₦	Atlanta GA
19.50	1.30	114	8	0	₦	Grande Prairie AB
5.79	0.44	69	5	1	₦	Miami FL
56.50	24.29	179	16	0	₦	Secaucus NJ
1.15	0.44	165	20	0	₦	New York NY
65.19	4.30	29	14	6	¥	Newbury Park CA
9.92	0.84	521	9	0	¥	Burlington VT
4.56	1.79	267	193	0	¥	Washington DC
0.26	0.30	1062	354	1	¥	New York NY

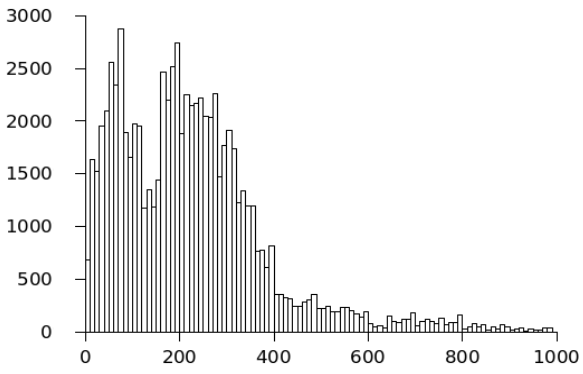

Figure 6. Prevalence / average delay (milliseconds) distribution graph of players.

After training with 70% of the data with fuzzy rule learner, our model is able to guess the remaining %30 number of disconnections with the accuracy of 81.057% this time.

Table 5. Confusion matrix

predefined \ prediction	online	offline
online	2802	550
offline	193	74

Only 267 of total rows are marked as worst-case class by using predefined arithmetic methods (a constant cut-off value) based on the interviews with network admins and gaming platform moderators, and analysis of existing tools. After training with 70% of the data with RProp learner, our model is able to guess the remaining 30% number of disconnections with the accuracy of 79.47% compared to predefined assessments. Confusion matrix is given in Table 5.

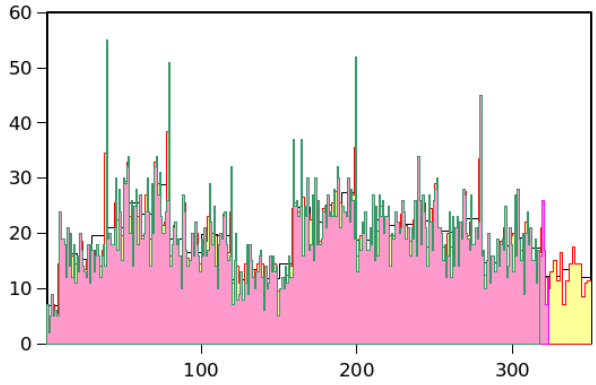

Figure 7. Comparison graph of predetermined mathematical method, fuzzy rule learner and Rprop learner.

Within the evaluation graph, shown in Figure 7, outputs from three neural network methods and predefined cut-off point based method are compared. Black line is a reference for actual data from the dataset with extreme values removed. Pink area is the output of evaluation that is executed with a predetermined cut-off point. This area is considered as 100% correct, hypothetically for the test. Yellow area is the extension of the pink area (which considered correct) where both methods fail to produce an expected result. The green area represents inconsistencies that belongs to the Rprop learner, the closest performer to conventional method. Values from this learner is also emphasized in graph with purple color in the areas where the difference between arithmetic methods is the most visible. (Also, the Rprop learner covers a larger area compared to other method). The red area on top of green/pink area represents inconsistencies (a small percentage of difference) that belongs to the fuzzy rule learner, the weaker performer of the two distinct methods.

6. Discussion and Related Work

Application of fuzzy logic shows that, it is completely possible to analyze network quality of the player by providing specific inputs. By measuring input variables mentioned, by calling into third party services or starting in house performance detection network infrastructure, it is convenient to classify players. Although gaming experience in lowest quality network classes are very low, it looks like much better option than disconnecting players. For players in this class, game developers may switch to specially crafted and more network friendly maps, game modes or even execute adaptive packet compression, by trading of between CPU power and network transfer rate. By smoothly mixing bandwidth, latency and packet loss, prevention of incorrect disconnection of players is carried out. This is obvious especially, when latency values of the players are very close to minimum ping quota. By also considering internet service provider synchronization results and actual transmission layer data, by calling forth external modules, players that possible to lose connection, are classified accordingly thus resulting much better entertainment experience.

With neural network learner concerned, it is clear that both back-propagation neural networks and mixed fuzzy rule formation shows similar and high accuracy results. Number of records were also not very high which makes learning process harder. Future work may include combination of fuzzy methods with neural networks and regression analysis. Consideration of the time zones is also a good candidate for future work. Time zone differences between players and high loads at ISP infrastructure in specific time intervals may be evaluated. Examples shows that, evaluating line conditions of a typical game player with a fast learner may be an option to decide without human interaction or solid threshold values. Therefore, mistaken disconnects because of latency spikes are much less; resulting much richer gaming experience. The test can also be adapted for occasions when the players themselves are hosts, and hence, pattern recognition methods decrease rate of host migrations.

Various techniques have been adopted to enrich the gaming experience. In this study, we have only covered techniques to detect anomalies within players and disconnect them. Other studies propose methods to optimize the game's itself to provide an acceptable medium even with players that have the worst networking conditions. These optimizations are highly coupled with game's genre and architecture. A perfect example would be the study by (Lee et al., 2005), that aims to find the most optimal server with an NP-hard algorithm. Our study in this paper aims to detect ones among all players to disconnect from the game session. However, it is possible to expand further and rank (as we have experienced with the fuzzy method) the players and mark one as the active server. Contrary to study in (Lee et al., 2005), the player's itself would be the server as opposed to games with a central server. And also, it may be possible to run the neural network algorithm on the gaming device itself. Most gaming devices has multiple processing units which may be very efficient, unless the game does not utilize the hardware completely. This type of games may be the platform (2D) games that sacrifice visual quality in terms of gameplay and rapid networking. Another study by (Cordeiro et al., 2007), shows that even well-engineered and well-known games utilize locking-based synchronization primitives when processing player clients. In the mentioned study, multiple cores of CPUs are more efficiently used by modifying some parts of the game's server. The study is further expanded in (Munro et al., 2014), with addition of load balancing into equation. Instead of some constant time quanta, threads are given variable running times. Strategy for this timing is one of Longest Processing Time, Shortest Processing Time, Round Robin or Shorted Round Robin. Effects of the latency on a typical strategy game is discussed in (Claypool, 2005). This study also explains usage of the TCP protocol within strategy games, contrary to most fast-paced games utilize UDP protocol. Benefits of utilizing prediction within clients and techniques about decreasing side effects of latency are discussed in the study by (Bernier, 2001).

7. Conclusion

Evaluating the player's network statistics and classifying players is a complex problem whereas type of network, delays caused by routers network filter should be considered. Depending on the game, players with high latency may slow down the entire mesh. Also, with games with client-server architecture, the cause of high latency may be originated from server's itself. It is imperative to initiate a load testing before measurement of input values. After the load testing, classifier, fuzzy and neural network methods are examined in different short sessions (15-20 minutes), ideally with the same players and the same game rules. Due to indeterministic nature of gaming sessions, it is very hard to reproduce the tests with exact same conditions. This leads to the insertion of neural networks into equation. A system that learns from previous sessions has much, much higher chance to adapt changing network environments. Changing conditions that originate from internet service providers and change in connection quality during rush hours can be coped with neural networks methods. On the other hand, fuzzy and machine learning methods are easier to implement and embed into existing online platforms. It is possible to simulate these methods with Matlab, thus initial investment is much less compared to neural network methods. A learned set is also not required, but lack of learnability makes these two methods best for human moderated systems. Neural network methods, on the other hand, can be completely independent and run with minimal observation. This shows us, all three methods that we have discussed has trade-offs. Complex methods do not necessarily yield the best gaming experience. Thus, selection of the most suitable method for the specific game type is crucial.

References

Bernier, Y.W., 2001. Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization, Proceedings of the Game Developers Conference, GDC '01, February 2001.

Chan, L., Yong, J., Bai, J., Leong, B. and Tan, R., 2007. Hydra: a massively multiplayer peer-to-peer architecture for the game developer, Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games, NetGames '07, September 19-20, 2007, Melbourne, Australia, p.37-42.

Chen, J., Wu, B., Delap, M., Knutsson, B., Lu, H. and Amza, C., 2005. Locality aware dynamic load management for massively multiplayer games, Proceedings of the Tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, PPoPP '05, June 15-17, 2005, Chicago IL, USA, p.289-300.

Chen, K.T., Huang, P., Huang, C.Y. and Lei, C.L., 2005. Game traffic analysis: an MMORPG perspective, Proceedings of the international workshop on Network and operating systems

support for digital audio and video, NOSSDAV '05, June 13-14, 2005, Stevenson WA, USA, p.19-24.

Claypool, M., 2005. The effect of latency on user performance in Real-Time Strategy games. Computer Networks, 49(1), 52-70.

Cordeiro, D., Goldman, A. and Silva, D.D., 2007. Load balancing on an interactive multiplayer game server, Proceedings of the 13th International Euro-Par Conference, Euro-Par '07, August 28-31 2007, Rennes, France, p.184-194.

Harcsik, S., Petlund, A., Griwodz, C. and Halvorsen, P., 2007. Latency Evaluation of Networking Mechanisms for Game Traffic, Proceedings of the Sixth Workshop on Network and System Support for Games, NETGAMES '07, September 19-20, 2007, Melbourne, Australia, p.129-134.

Henderson, T., 2001. Latency and user behaviour on a multiplayer game server, Proceedings of the Third International COST264 Workshop on Networked Group Communication, NGC '01, November 07-09, 2001, London, UK, p.1-13.

Joe, I., 1996. Packet loss and jitter control for real-time MPEG video communications. Computer Communications, 19(11), 901-914.

Lebres, I., Rita, P., Moro, S. and Ramos, P., 2018. Factors determining player drop-out in Massive Multiplayer Online Games. Enterteinment Computing, 26, 153-162.

Lee, B. and Chang, J., 2016. Packet loss concealment based on deep neural networks for digital speech transmission. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 24(2), 378-387.

Lee, K., Ko, B. and Calo, S., 2005. Adaptive server selection for large scale interactive online games. Computer Networks, 49(1), 84-102.

Mohanty, R., Ravi, V. and Patra, M.R., 2010. Web-services classification using intelligent techniques. Expert Systems with Applications. 37(7), 5484-5490.

Munro, J., Appiah, K. and Dickinson, P., 2014. Investigating informative performance metrics for a multicore game world server. Enterteinment Computing, 5(1), 1-17.

Reid, N.P. and Seide R., 2002. Wi-Fi (802.11) Network Handbook, First edition, Berkeley CA, USA, McGraw-Hill Osborne Media.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning internal representations by error propagation. Parallel distributed processing: explorations in the microstructure of cognition (MIT Press), Volume 1, July 1986, Cambridge MA, USA, p.318-362.

Shanahan, J.G., 2000. Soft Computing for Knowledge Discovery: Introducing Cartesian Granule Features, First edition, Meylan, France, The Springer International Series in Engineering and Computer Science.

URL-1, 2004. Voice and Video Enabled IPSec VPN Solution Reference Network Design, Cisco Systems, http://www.cisco.com/application/pdf/en/us/guest/netsol/ns171/c649/ccmigration_09186a008074f2d8.pdf

URL-2, 2005. Star Wars® Battlefront IITM Server Manager Users Guide, Black Bag Operations and Lucasfilm Ltd., https://hiddenarrows.com/sites/default/files/utilities/SWBF2_BBO_SM-RM_User_guide.pdf

Zander, S., Leeder, I. and Armitage, G., 2005. Achieving Fairness in Multiplayer Network Games through Automated Latency Balancing. Proceedings of the ACM SIGCHI International Conference on Advances in computer entertainment technology, ACE '05, June 15-17 2005, Valencia, Spain, p.117-124.