Dynamic Adaptive Communication Strategy for Fully Immersive, Interactive and Collaborative Virtual Reality Applications

Adjeryan Cartaxo Freitas[1], Diego Roberto Colombo Dias[2], Alexandre Fonseca Brandão[3], Rita de Fátima Rodrigues Guimarães[1,4],and Marcelo de Paiva Guimarães[1,5(✉)]

[1] Computer Science Master Program (Unifaccamp), Campo Limpo Paulista, Brazil
adjeryance@gmail.com, rita.guimaraes@gmail.com, marcelodepaiva@gmail.com
[2] Computer Science Department (UFSJ), São João Del Rei, Brazil
diegocolombo.dias@gmail.com
[3] Institito de Física Gleb Wataghin (UNICAMP), Campinas, Brazil
abrandao@ifi.unicamp.br
[4] Applied Linguistics (Institute of Language Studies/Unicamp), Campinas, Brazil
[5] Núcleo de Ensino a Distância (UAB/Reitoria), UNIFESP, São Paulo, Brazil

Abstract. An online meeting allows people with a common purpose to discuss ideas, goals, and objectives, regardless of national and international boundaries. Using internet technologies, users can communicate with others at remote locations without leaving their offices. However, internet communication can show considerable variation in terms of packet loss, time variation, and delay. This paper presents a strategy for dynamically adapting communication between fully immersive, interactive, and collaborative virtual reality applications. Our strategy is implemented based on the optimal path forest classifier, allowing us to analyze the network communication and to provide parameters for the application communication library to perform dynamic adaptation. We also present the GClassifier tool, which is used to implement this strategy, and some test results.

Keywords: Virtual reality · Network · Communication adversity · Classifier

1 Introduction

Fully immersive, interactive, and collaborative virtual reality applications allow users to interact and navigate in synthetic worlds (i.e., both real and virtual) while solving a single task. They also include multi-projection applications (e.g., cave automatic virtual environments (CAVEs) and power walls). These systems are composed of diverse virtual reality environments that are connected remotely with each other over the internet. In the early years, these applications were executed using high-end supercomputers; however, due to advances in computer hardware and software, the graphic cluster has become the default architecture used for these applications [1–4].

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12255, pp. 771–783, 2020.
https://doi.org/10.1007/978-3-030-58820-5_55

Each instance of the application has a local graphic cluster that renders multiple views of the same visual dataset in real time, and each cluster may be composed of different types of nodes (e.g., rendering nodes, a file server node, input nodes, and a master node to manage the environment), while intra-cluster communication uses a local area network (LAN) to handle data from multiple nodes in real time using a master/slave architecture to run a local environment. Inter-cluster communication is achieved by using a peer-to-peer approach to connect the remote environments, and an intelligent network service is required to interconnect these resources. One of the development challenges of these environments is to provide efficient communication and synchronization that can guarantee intra- and inter-cluster datalock (i.e. ensuring data coherence between processes) and framelock (i.e. maintaining the coherence of scenes).

A dynamic network protocol (e.g., Transmission Control Protocol (TCP), Datagram Protocol (UDP) or Stream Control Transmission Protocol (SCTP)) can be used to implement inter-cluster communication (via a wide area network or WAN), allowing for adaptation of communication and synchronization according to the situation that is faced. Inter-cluster communication uses LANs that can be considered close to perfect. A fully collaborative virtual reality application may generate large communication datasets, since the frame rate must be above 60 frames per second (fps) to give a good stereoscopic view. Although there are several sniffers that can assist in network analysis, such as TCPdump[1], Ngrep[2], and Snort[3], there is a lack of tools that can classify the data collected in this way. Sniffers do not take advantage of the human cognitive capacity of learning and pattern recognition, and the network administrator typically performs a manual analysis of a massive amount of data. Some efforts have already made using approaches such as support vector machines (SVMs) [5] and artificial neural networks [6], although these require considerable computational time for large datasets, especially in the training phase [7].

This paper presents a strategy for analyzing packet loss variables (packets lost in the connections between nodes), time variation (average time for packet delivery between nodes), and delay (delay in packet delivery between nodes). This allows a communication library to perform dynamic adaptation, which is used to implement fully immersive, interactive, and collaborative virtual reality applications. This forms the main contribution of this work, which proposed a powerful real-time solution for network adversity analysis. We use a supervised pattern recognition technique based on the optimum path forest (OPF) [8] to classify the adversity variables. Based on the requirements of fully immersive, interactive, and collaborative virtual reality applications, a dataset was manually classified into several adversity classes. The resulting set associated with a specific description was used to train the classifier. We tested our approach using a dataset from a traffic simulation, and compared the predicted classes with the adversity classes initially defined. We also compared the captured data using a network tool to provide adaptation parameters for the communication libraries. Also in

[1] https://www.tcpdump.org/manpages/tcpdump.1.html.

[2] http://ngrep.sourceforge.net/usage.html.

[3] https://www.snort.org/.

this paper, we present a graphical tool called GClassifier, which can be used to automate the training and classification tasks via a friendly interface.

This paper is organized as follows. In Sect. 2, we discuss the features of fully immersive, interactive, and virtual reality applications. Section 3 explains the details of the GClassifier tool used to implement our strategy. Section 4 presents some test results. Finally, Sect. 5 contains the conclusion of this paper.

2 Fully Immersive, Interactive and Collaborative Virtual Reality Applications

Fully immersive, interactive, and collaborative virtual reality applications allow remote users to share a space at the same time. Their main features are immersion, interactivity, involvement, and collaboration. Immersion provides the user with the perception of being physically present inside the simulated world, and can be achieved, for instance, using 3D stereoscopic displays and motion tracking devices. Interactivity is associated with the responsiveness of the simulated environment to user actions, while involvement is related to the user's engagement with the virtual environment. Collaboration in a virtual reality environment [9, 10] allows groups to work together on the execution of a single task. In this paper, we focus on virtual reality systems that accommodate participants that are connected remotely.

Today, it is possible to create high-quality virtual reality applications using high-level tools such as Unity 3D[4] and Ogre3D[5], which allow sophisticated synthetic environments to be built. However, to enable collaboration, it is necessary to consider that during navigation and interaction, users perform actions that are transmitted over the network [11], and the remote site will receive updates caused by these actions after a certain amount of delay [12]. This problem has led to the use of low-level application programming interfaces (APIs) (e.g., sockets over TCP/UDP, PVM[6], and MPI[7]) to facilitate information exchanges and synchronization. Thus, the applications running on top of these lower-level APIs become limited by factors such as network performance. Input and output devices (e.g., motion tracking, audio, and graphics rendering) may also cause additional latency in these applications, although this is beyond the scope of the present paper.

The graphics clusters used by fully collaborative virtual reality applications give multiple views of the same visual dataset. The nodes in each local cluster access the entire dataset, and then independently determine how much of the dataset is visible given the assigned viewing frustum, based on the local user view, and render only that part. The challenge is to provide a coherent, seamless, and continuous display using the isolated, distributed visual nodes in each local environment. In these applications, data ranging from the view frustum to geometric primitives or even avatar positions are

[4] https://unity.com/.

[5] https://www.ogre3d.org/.

[6] http://pvm-plus-plus.sourceforge.net/.

[7] https://www.open-mpi.org/.

continuously sent between nodes, and rendering must be synchronized between the intra- and inter-cluster nodes.

Despite the existence of libraries that can assist in the development of cluster applications, a method has not yet been proposed for maintaining quality data exchanges in an inter-cluster scenario, and this is a promising line of research.

3 GClassifier

Figure 1 depicts the strategy workflow, which includes training, testing, and a run-time network evaluation, to allow the library to adapt the communication to meet the requirements of the application. The process starts by defining the network requirements of the full, immersive, interactive, and collaborative virtual reality applications. The classes and their features are then (manually) defined. Next, the training set is automatically created, by capturing data using a network simulator (e.g., Network Emulation or NetEm)[8]. This dataset is saved and used for the internal classification of the ensemble. Finally, it is ready to be used directly at runtime by the library to adapt the network communication. The library must be tailored to capture packets, trigger the classification and adapt the communication during runtime (e.g., to choose another network protocol, change the network topology used, and/or perform data buffering if the bandwidth is not sufficient).



Fig. 1. Steps in the proposed strategy for network communication adaptation at application runtime

Figure 2 illustrates the architecture of the software developed here. The GClassifier tool is used to implement this scheme. The main idea is to check the underlying communication in terms of packet loss, time delay, and time variation, and the system was coded using the Java language. GClassifier provides a graphical interface that allows the network administrator to fill in the parameters to be used during the training (e.g., physical network card, percentage packet loss and network delay) and test phases (e.g., number of captured packets). It uses charts to represent the results of network communication analysis. The core of our strategy is the OPF framework, which allows the user to build classifiers by changing their modules (e.g., the adjacency relation, prototype estimation methodology, and path-cost function). It also supports multi-classes and

[8] http://man7.org/linux/man-pages/man8/tc-netem.8.html.

can handle some degree of overlap between classes [8]. Our solution is based on a supervised classifier that uses a complete graph as an adjacency relation. The training set is interpreted as the nodes of a graph in which the arcs are defined by a given adjacency relation and weighted by the distance function (fmax, which assigns the maximum arc weight along the path). It contains samples from different classes, each of which has a set of features and a distance function that are used to measure their dissimilarity in the feature space. The OPF aims to assign a correct class label to each new sample (prototype), and these compete among themselves to conquer the remaining samples. This process results in optimum path trees. The classes were defined manually based on the network requirements for fully immersive, interactive, and collaborative virtual reality applications. To build the training set, we added values for delay and packet loss for outgoing packets from a specific network interface using the traffic control tool NetEm, and used the ping tool to test the reachability of each node.

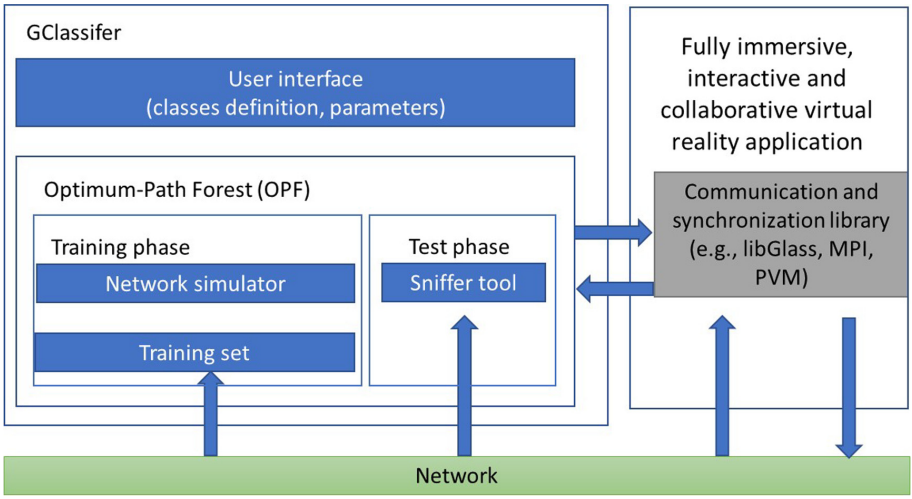

Fig. 2. Architecture of the GClassifier tool

Figure 3 shows an example of the results of a reachability test using the ping tool. The ping results are compared with the optimum path tree, and the tree that matches the sample is associated with the class (rotuled). The communication application library can then gather packets from the network communication, and classify and mitigate the existing adversities.

According to Chen [13], the user experience of a desktop collaborative virtual reality environment is affected when the network latency is greater than 200 ms and/or packet loss is more than 60%. Jitter is the most harmful effect, since it is necessary to maintain a constant speed for the delivery rate of packets; however, this can be mitigated using buffering techniques [14].

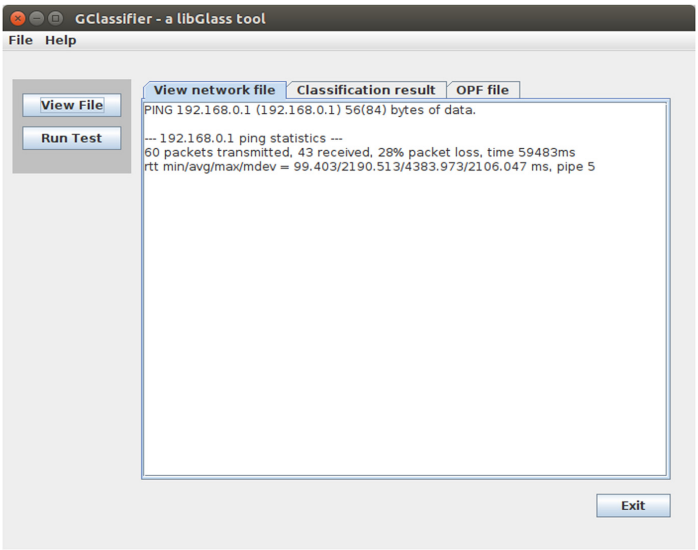

Fig. 3. Example from GClassifier showing the results of a node reachability test

Table 1 lists the range of values defined for each input variable (packet loss, time variation, and delay) and shows the consequences (not harmful, harmful, or very harmful). These values were based on a systematic review performed by Singhal and Zyda [15]. The attributes shown in the table were based on the minimum expected requirements for a graphics application (rendering) with 60 fps, in which the information for each frame is transmitted in one message, resulting in 60 messages per second. The three attributes considered are packet loss, time variation, and delay. Ling et al. [16] reported that delays of up to 200 ms are detrimental for collaborative applications. This also applies to virtual reality applications, due to the real-time interactions between users, and since environmental elements are updated based on each user's actions and reactions. Chen [13] found that packet loss is altered in an

Table 1. Range of values defined for each input variable

	Packet loss	Time variation	Delay
Not harmful	0–3%	0–5 ms	0–60 ms
Harmful	2–6%	4–10 ms	55–200 ms
Very harmful	5–20%	9–100 ms	190–600 ms

environment where the application achieves at least 40 status updates per second. These results allowed us to define the parameters for each OPF class. These values can be changed according to specific requirements; for example, a stereoscopic application requires more fps than a non-stereoscopic one.

Each input variable is used to measure a certain problem associated with network communication, and is related to the network communication adversity. Table 2 shows a combination of 27 classes that are used to define the type of dynamic adaptation

Table 2. Classes of adversity in network communication

Class	Packet loss	Time variation	Delay	Class	Packet loss	Time variation	Delay
1	Not harmful	Not harmful	Not harmful	15	Harmful	Harmful	Very harmful
2	Not harmful	Not harmful	Harmful	16	Harmful	Very Harmful	Not harmful
3	Not harmful	Not harmful	Very harmful	17	Harmful	Very Harmful	Harmful
4	Not harmful	Harmful	Not harmful	18	Harmful	Very Harmful	Very harmful
5	Not harmful	Harmful	Harmful	19	Very harmful	Not harmful	Not harmful
6	Not harmful	Harmful	Very harmful	20	Very harmful	Not harmful	Harmful
7	Not harmful	Very harmful	Not harmful	21	Very harmful	Not harmful	Very harmful
8	Not harmful	Very harmful	Harmful	22	Very harmful	Harmful	Not harmful
9	Not harmful	Very harmful	Very harmful	23	Very harmful	Harmful	Harmful
10	Harmful	Not harmful	Not harmful	24	Very harmful	Harmful	Very harmful
11	Harmful	Not harmful	Harmful	25	Very harmful	Very harmful	Not harmful
12	Harmful	Not harmful	Very harmful	26	Very harmful	Very harmful	Harmful
13	Harmful	Harmful	Not harmful	27	Very harmful	Very harmful	Very harmful
14	Harmful	Harmful	Harmful				

required from the communication library. The most representative instance of each class in the training set was chosen to be in the border region between the classes. Each class is expected to have an associated action from the communication library. This allows the master nodes of each local cluster to regulate its sending rate based on the class defined in a unified approach.

After the training and testing phases, the GClassifier is ready to classify new data using the categories of network communication adversity shown in Table 2. The application communication library can then mitigate the connection issues by applying the following actions:

- Altering the communication protocol: This refers to the set of rules used by computers to exchange messages with each other. TCP is often the default protocol, although in case of adversity, it can be combined with UDP; for example, TCP can be used for critical packets while UDP is used for status update packets. The library could also switch to SCTP, which is a reliable transport protocol that operates over an unreliable and unconnected packet service, such as Internet Protocol (IP).
- Buffering data: Buffers can be used to reduce packet loss as they can compensate for bursts of traffic in which routers cannot handle message forwarding at a given instant [17]. Buffers can also be used to reduce the variability of sender nodes and to transform a variable receiving rate into a constant rate. For example, environmental state update packets can be buffered by a sender node in order to maintain a constant frame rate.
- Changing the network packet size: The use of large packets means that the sender needs to preallocate buffers that are large enough to send and receive the maximum possible size, while the use of small packets will involve more messages requesting to send data.
- Predicting packets: When the adversities are related to packet loss, the communication library can apply a linear prediction model in which future packet signals are estimated using a linear function based on previous samples. A latency compensation method can be used to reduce network latency. One example of a technique that can be adopted is dead reckoning, which performs extrapolation based upon the data received.

Our approach suggests solutions based on the classes identified by the classifier, and considers the actions required to mitigate these adversities. For example, if packet loss is “not harmful”, time variation is “very harmful”, and delay is “harmful”, the communication library may try to mitigate these issues using TCP combined with UDP.

4 Tests

GClassifier automates the training, classification, and visualization of results, and the configuration of the tests is carried out via its interface. The interface contains parameters such as the classes of adversities, and parameters to be passed to NetEm (e.g., the number of packets sent and the delay). We performed 10 tests. Figure 4 illustrates the results of Test 3, in which 60 packets were sent and a delay of 6 ms, a

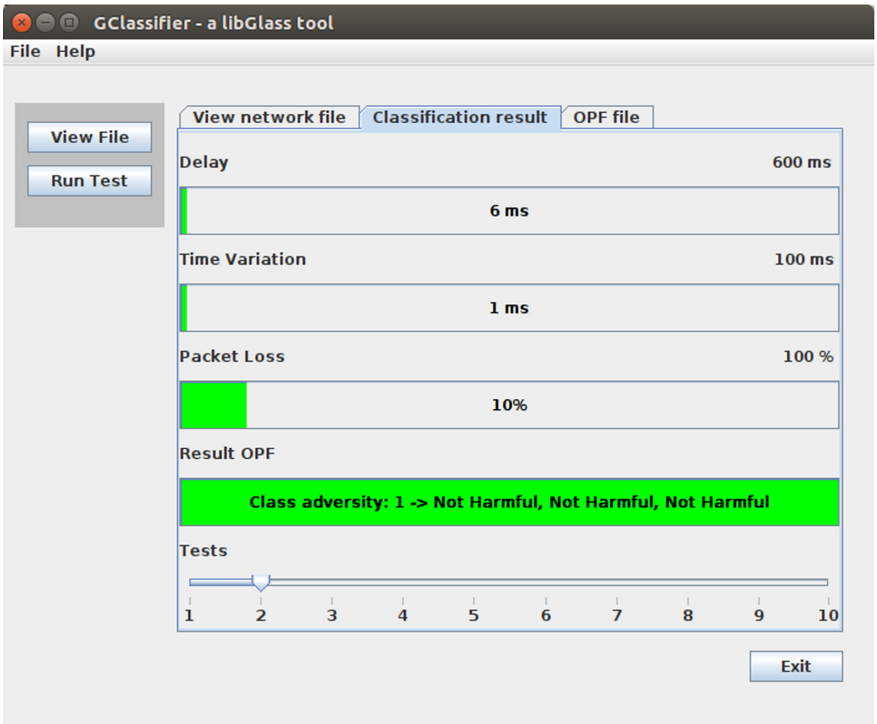

Fig. 4. Reachability test in which no problems were detected

time variation of 1 ms, and a packet loss of 10% were simulated using NetEm.

GClassifer defined this scenario as Class 1 (delay not harmful, time variation not harmful and packet loss not harmful). No adversity was found when a reachability test was executed.

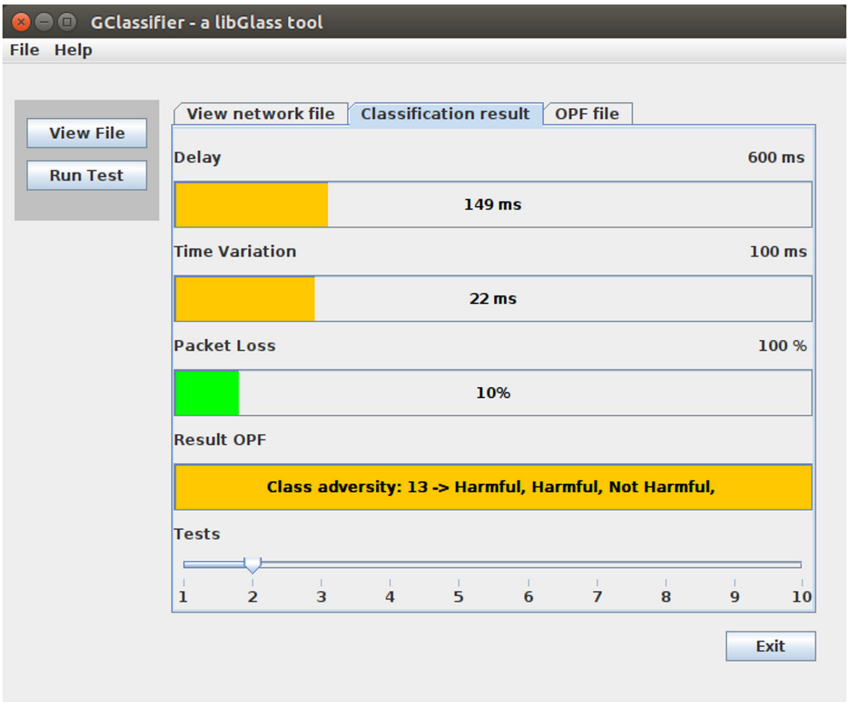

Fig. 5. Reachability test in which a problem was detected

Figure 5 illustrates the results of Test 8, in which a harmful level of adversity was found. This was classified as Class 13. During this test, 60 packets were sent, and a delay of 149 ms, a time variation of 1 ms, and a packet loss of 10% were simulated using NetEm.

The results of Test 3 are shown in Fig. 6. This scenario was classified as Class 27. During this test, 60 packets were sent, and a delay of 501 ms, a time variation of 87 ms, and a packet loss of 60% were simulated using NetEm.

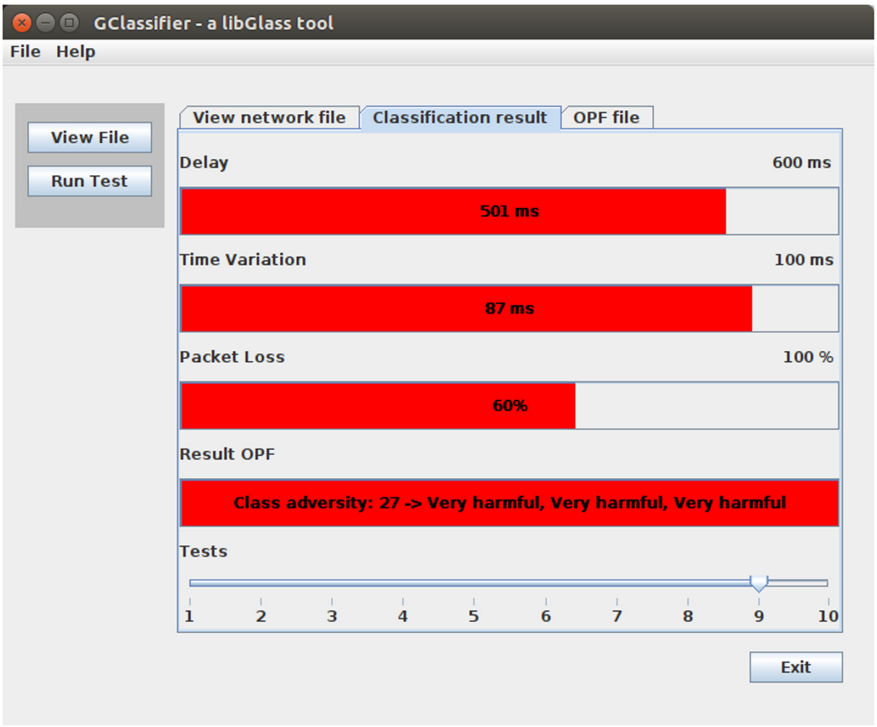

Fig. 6. Reachability test in which a serious problem was detected

5 Conclusion

Fully immersive, interactive, and collaborative virtual reality applications aim to use human sensory perception to send information to users' brains, to help them to solve a given task within a computer-generated environment. The developers of these applications need to overcome many barriers in terms of combining different hardware and software to achieve a sense of presence.

In recent years, due to the advances in computer hardware, software, and networking, these applications have been developed using graphics clusters, which require a knowledge of diverse areas such as computer graphics, human-computer interfaces, distributed systems, and computer networking. Communication within these applications is typically implemented using high-level functionality on top of the low-level primitives that are available (e.g., BSD sockets, PVM, and MPI). One example of a high-level library is libGlass[9], which is tailored for fully immersive applications based on intra-cluster communication. This library offers primitives that can maintain synchronicity and coherence over the data, and provide datalock and framelock.

[9] http://libglass.sourceforge.net/.

Currently, the most common solutions are based on computer graphics clusters, with each environment implementing its own. These exchange data and carry out synchronization using inter-cluster communication over the internet. This makes it necessary to conform to strict timing dependencies, meaning that network communication is likely to experience bottlenecks caused by problems such as packet loss, delay, and time variation. We created a strategy to analyze the variables of packet loss, time variation, and delay to allow dynamic adaptation of communication within fully immersive, interactive, and collaborative virtual reality applications. Our strategy was based on the use of a supervised pattern recognition technique called OPF to classify these variables into 27 classes of adversities. To train our classifier, we used data collected by the simulator NetEm. The GClassifier tool automates the training and test phases. The results can be used directly by any library to adapt network communication in the appropriate way, for example, by changing the buffer size and/or the protocol used. In future, we plan to test other pattern recognition techniques.

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Kuhlen, T.W., Hentschel, B.: Quo vadis cave: does immersive visualization still matter? IEEE Comput. Graph. Appl. **34**, 14–21 (2014). https://doi.org/10.1109/MCG.2014.97
2. Tredinnick, R., Boettcher, B., Smith, S., Solovy, S., Ponto, K.: Uni-CAVE: A Unity3D plugin for non-head mounted VR display systems. In: 2017 IEEE Virtual Reality (VR). pp. 393–394 (2017). https://doi.org/10.1109/VR.2017.7892342
3. Chung, H., Andrews, C., North, C.: A survey of software frameworks for cluster-based large high-resolution displays. IEEE Trans. Vis. Comput. Graph. **20**, 1158–1177 (2014). https://doi.org/10.1109/TVCG.2013.272
4. Luo, L., Mo, J., Chen, H.: Multi-channel visual reality system based on computer cluster. In: 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). pp. 1–6 (2017). https://doi.org/10.1109/M2VIP.2017.8211453
5. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. pp. 144–152. Association for Computing Machinery, New York, NY, USA (1992). https://doi.org/10.1145/130385.130401
6. Aggarwal, C.C.: Neural Networks and Deep Learning: A Textbook. Springer, Cham (2018)
7. Papa, J.P., Falcão, A.X., Victor, H.C.: Efficient supervised optimum-path forest classification for large datasets. Pattern Recogn. **45**, 512–520 (2012). https://doi.org/10.1016/j.patcog.2011.07.013
8. Papa, J.P., Falcão, A.X., Suzuki, C.T.N.: Supervised pattern classification based on optimum-path forest. Int J Imag Syst Tech. **19**, 120–131 (2009). https://doi.org/10.1002/ima.20188
9. Nguyen, T.T.H., Duval, T.: A survey of communication and awareness in collaborative virtual environments. In: 2014 International Workshop on Collaborative Virtual Environments (3DCVE). pp. 1–8 (2014). https://doi.org/10.1109/3DCVE.2014.7160928

10. Bente, G., Rüggenberg, S., Krämer, N.C., Eschenburg, F.: Avatar-mediated networking: increasing social presence and interpersonal trust in net-based collaborations. Hum. Commun. Res. **34**, 287–318 (2008). https://doi.org/10.1111/j.1468-2958.2008.00322.x
11. Shirmohammadi, S., Georganas, N.D.: An end-to-end communication architecture for collaborative virtual environments. Comput. Net. **35**, 351–367 (2001). https://doi.org/10.1016/S1389-1286(00)00186-9
12. Gutwin, C., Benford, S., Dyck, J., Fraser, M., Vaghi, I., Greenhalgh, C.: Revealing delay in collaborative environments. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 503–510. Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/985692.985756
13. Ling, C.: Effects of network characteristics on task performance in a desktop CVE system. In: 19th International Conference on Advanced Information Networking and Applications (AINA 2005) (AINA papers). vol. 1, pp. 821–826 (2005). https://doi.org/10.1109/AINA.2005.171
14. Anthes, C., Haffegee, A., Heinzlreiter, P., Volkert, J.: A scalable network architecture for closely coupled collaboration. Comput. Inf. **24**, 31–51 (2012)
15. Singhal, S., Zyda, M.: Networked Virtual Environments: Design and Implementation. Addison-Wesley Reading, Boston (1999)
16. Chen Ling, X.X., Chen Gen-Cai, C.C.: An effective communication architecture for collaborative virtual systems. In: International Conference on Communication Technology Proceedings, ICCT 2003. vol. 2, pp. 1598–1602 (2003). https://doi.org/10.1109/ICCT.2003.1209833
17. Sequeira, L., Fernández-Navajas, J., Casadesus, L., Saldana, J., Quintana, I., Ruiz-Mas, J.: The influence of the buffer size in packet loss for competing multimedia and bursty traffic. In: 2013 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS). pp. 134–141 (2013)