An Optimised Design for Streaming Media in Cloud Desktop System

Wei-Chen Ma
Southwest Jiaotong University, Chengdu, Sichuan,China
mwcnmgg@163.com

Abstract—As the earliest and most classic cloud desktop protocol, VNC has been limited in its usage scenarios and further development due to its lazy update mechanism and outdated coding methods, which fail to meet the requirements for video playback and other demands. This paper proposes an optimization approach for VNC systems by dynamically adjusting the parameters based on network latency to control the minimum communication time between the client and server, allowing continuous data updates during this period. To address the issues related to outdated coding methods and low bandwidth requirements, a novel algorithm is introduced that utilizes motion vectors to differentiate between high and low motion areas in video frames. High motion areas are encoded and decoded using H.264, while low motion areas maintain the coding and decoding method of the RFB protocol. This approach not only ensures lower bandwidth usage but also provides a satisfactory user experience.

Keywords- cloud desktop system; vnc; lazy update;video coding; high motion; rfb;

1. INTRODUCTION

With the continuous advancement of computer technology and the relentless progress in hardware capabilities, the demand for remote desktop technology has witnessed a significant surge [1]. The proliferation of remote work has established it as a prevalent and widely adopted mode of work worldwide. This discernible upswing in remote work requirements has compelled numerous companies and organizations to embrace remote desktop technology as a means to facilitate their employees' remote or work-from-home setups. The remote desktop protocol is at the core of remote desktop technology, which serves as the linchpin for seamless connectivity and interaction. Among the well-known remote desktop protocols in existence, namely RDP, SPICE, and VNC, each possesses its distinct design principles, features, and suitability for specific scenarios [2,3].

VNC, being one of the pioneering remote desktop systems, comprises three integral components: VNC Server, VNC Client, and the RFB protocol acting as the communication conduit between them. The VNC Server harnesses the actual computational resources of the host machine to perform an array of tasks encompassing computation, rendering, compression, and transmission of image data. On the other hand, the VNC Client assumes the responsibility of receiving the transmitted image data, decoding the associated information, and rendering it locally for display. The RFB protocol governs the seamless exchange of communication and interaction between the VNC Server and the VNC Client [4]. Leveraging this architectural framework, VNC empowers users to remotely access and assume control over remote computers via lightweight thin clients across the network, thereby enabling users to seamlessly tap into remote resources across disparate operating systems and devices, catering to their fundamental office requirements. However, the evolving nature of remote work demands transcends the realm of simplistic office tasks, extending into complex realms such as comprehensive office workflows, entertainment, education, and more. VNC in its standalone form falls short of meeting these evolving demands, leading to the attention of computer professionals being drawn towards augmenting and elevating the capabilities of VNC systems.

2. RELATED WORK

2.1. Update Mechanism

By default, VNC uses a lazy update polling mechanism, as shown in Figure 1. In this mechanism, the server only responds when it receives a request from the client, which prevents unnecessary updates. However, this approach can have a significant impact on-screen updates in high-latency networks or scenarios with high update requirements. The problems caused by the lazy update mechanism are described in [5]. Limited bandwidth or unstable network conditions can lead to increased latency and response time. In reference [6], C. Taylor and J. Pasquale applied a message accelerator that acts as an intermediary between the VNC client and server, forwarding the information exchange between them. The accelerator requests updates more frequently from the server and forwards the screen update information to the client. In reference [7], a server-push mode is proposed as an alternative to the client-pull mode.

2.2. Encoding Algorithms

As for the encoding methods in VNC systems, RRE, CoRRE, ZRLE, and Hextile are commonly used. These encoding methods are lossless compression techniques with low compression ratios, which are not suitable for video transmission. In reference [8], a two-dimensional graphic terminal interface encoding method is mentioned, which achieves a better compression ratio and decoding efficiency when processing plain text files in word applications. However, this algorithm has a low compression ratio when processing video images. To address the limitations of VNC systems in handling video, gaming, and other scenarios, a new encoding method is proposed as a replacement for the

979-8-3503-0247-9/23/$31.00 ©2023 IEEE
DOI 10.1109/ISSSR58837.2023.00063

low-compression-rate encoding methods in the RFB protocol, as discussed in reference [9]. However, this approach imposes a high network load as the VNC system continuously uses the new encoding method at any given time. In reference [10], the concept of dividing video frames into high-motion and low-motion regions is introduced, and different encoding methods are applied separately to each region. [11] proposes an integrated remote control model that incorporates audio/video isolation technology, aiming to support separate multi-A/V sessions and improve the quality of multimedia applications.

Overall, these studies propose various approaches to enhance the encoding and transmission efficiency of VNC systems, considering factors such as network conditions, compression ratio, and the distinction between high-motion and low-motion regions in video frames.

3. System Design

3.1. Prototype System Implementation

The implementation of a prototype serves primarily as a reference to evaluate the proposed methods. The prototype is built on a desktop-level operating system platform such as Windows 10. It is developed based on the open-source "Tight VNC," which does not include the H.264 codec by default. Therefore, it requires porting the open-source audio and video codec FFmpeg to the system. In the subsequent text, the H.264 algorithm will be referenced, so a brief introduction to H.264 is provided. H.264 is a widely used video compression standard. It was jointly developed by the International Telecommunication Union (ITU) and the International Organization for Standardization (ISO) and was released in 2003. H.264 aims to provide efficient video compression while maintaining high video quality. The H.264 algorithm achieves compression by exploiting both spatial and temporal redundancies in video data. Spatial compression reduces redundancy within individual frames by using techniques such as intra-prediction, transform coding, and quantization. Intra-prediction predicts pixel values within a frame based on neighboring pixels, transform coding converts the predicted values into frequency domain coefficients, and quantization further reduces their precision, achieving compression. Temporal compression leverages redundancies between consecutive frames through inter-prediction, motion compensation, and variable-length coding. Inter prediction analyzes motion between frames and predicts pixel values by referencing previously encoded frames. Motion compensation compensates for the predicted motion by transmitting only the differences between predicted and actual frames. Variable-length coding efficiently encodes the residual data. H.264 also incorporates advanced features like multiple reference frames, adaptive quantization, and deblocking filtering to enhance compression efficiency and video quality. It has become the most widely adopted video compression standard due to its ability to deliver high-quality video at lower bit rates, making it suitable for various applications such as video streaming, broadcasting, video conferencing, and storage [12].

3.2. Video Codec Integration

The VNC server is typically a thin client, and the resource usage of the VNC server is strictly limited. When playing videos or games, the screen image updates rapidly. At this time, the complexity and compression ratio of the encoding method used by the server to encode the image information becomes important factors affecting the efficiency of the VNC system. The original VNC encoding method has low complexity but also a low compression ratio. This results in network congestion when using low compression ratio data in the mentioned scenarios. Therefore, it is advisable to consider both encoding time and compression ratio when selecting a new compression algorithm. According to the data in [13], it is easy to choose the H.264 encoding algorithm. This algorithm occupies the least amount of bitrate at the same YUV-PSNR and achieves the highest YUV-PSNR at the same bitrate. As mentioned in the previous section, FFmpeg is integrated into the proposed VNC system, and this encoder provides various encoding methods, including H.264, for future use.

3.3. Optimized Update Mechanism

In contrast to the original VNC system, the proposed system addresses the issues caused by the lazy update polling mechanism with a new update mechanism, as shown in Figure 2. Both the VNC Server and VNC Client are equipped with buffers. Once the server receives a FramebufferUpdateRequest from the client, it clears the buffer and sends a predetermined duration of update data until the timer expires or the buffer becomes full. The server continuously sends data from the buffer to the client until the connection is terminated or the buffer is empty.

To further optimize the system, a module can be added to the server to periodically obtain network latency information after establishing the VNC connection. Based on this latency information, the preset time for updates can be dynamically adjusted. When the preset time is set close to the network latency, the system can achieve better performance.

By implementing this new update mechanism and dynamically adjusting the preset time based on network latency, the system aims to overcome the limitations of the default VNC lazy update polling mechanism. It ensures more timely and efficient updates between the VNC Server and Client, resulting in improved user experience and reduced impact from high-latency network conditions.

3.4. Block-based motion estimation

Motion estimation is a critical technology in video coding used to estimate the motion information between video frames [14]. Its main purpose is to determine the displacement of each pixel between consecutive frames, providing a basis for subsequent video compression. One commonly used method for motion estimation is block-based motion estimation, which involves dividing the current frame into several blocks and searching for the best matching position in the reference frame for each block. This process calculates the motion vectors between blocks and offers various search algorithms.

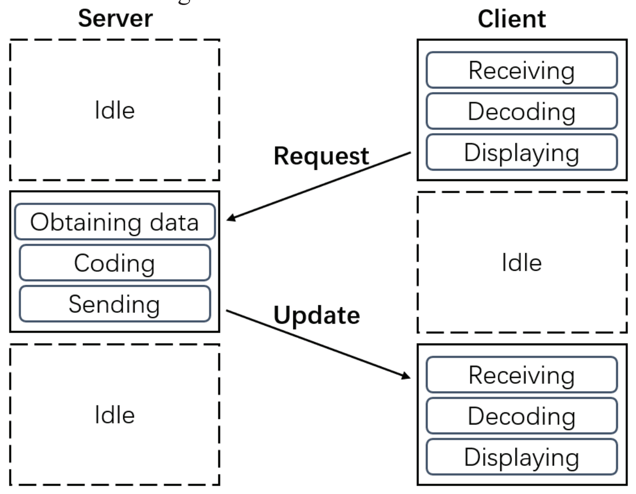

Figure 1. VNC lazy update mechanism

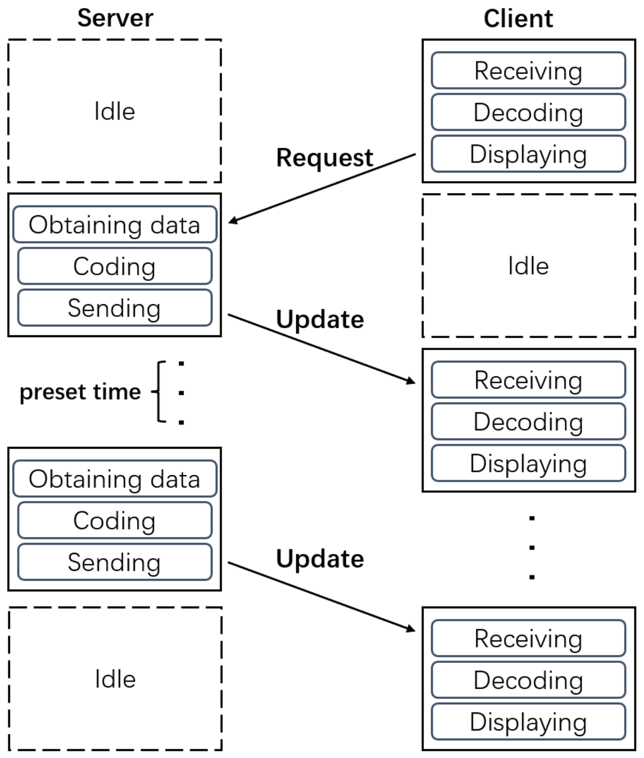

Figure 2. Optimized update mechanism

Block-based motion estimation is a simple, yet effective technique widely applied in video coding. Although not supported by the VNC system, it has provided inspiration for subsequent algorithm proposals. The functionality can be achieved through targeted development. For example, in the VNC system, consecutive image frames are typically differentially encoded by comparing and transmitting only the pixels that have changed from the previous frame. Based on this approach, the algorithmic idea of motion estimation can be introduced. By comparing each frame with the previous frame and calculating pixel differences, a motion estimation algorithm can estimate the motion vectors of each block, representing the block's displacement in the image. The specific utilization will be discussed in the next section.

3.5. Detection and Separate Transmission of High/Low Motion Regions

A. Detection of High/Low Motion Regions

First, upon receiving the FramebufferUpdateRequest at the server side and starting the processing of the screen data, the current frame is divided into several equally sized small blocks. The size of the blocks is typically determined by a trade-off between specific application requirements, desired precision, and computational complexity. The previous frame is chosen as the reference frame for comparison.

For each block in the current frame, a search is performed in the reference frame to find a similar block that best matches it. The matching process involves comparing the blocks using a metric such as the sum of squared differences (SSD) to determine the best-matched block. The motion vector is then computed based on the displacement between the current block and its best-matched block in the reference frame. The calculation of motion vectors is as follows:

$$SSD(dx,dy) = \sum(C(i,j) - R(i+dx, j+dx))^2 \quad （1）$$

$C(i,j)$ indicates the coordinates of a pixel in the current block，$R(i+dx, j+dx)$ indicates the corresponding pixel in the reference block after applying the displacement (dx, dy).

$$MV = argmin(SSD(dx,dy)) \quad （2）$$

“argmin” is a function that indicates the value of the independent variable that minimizes a function and achieves the minimum value.

B. Partition Encoding and Sending

After obtaining the motion vectors, the high and low motion regions can be determined based on the magnitude of the corresponding block's motion vector. The high motion regions are then encoded using the H.264 encoder, while the low motion regions are encoded using the default VNC encoder. The encoded data is stored in their respective buffers for data transmission.

For the low motion regions, the data is transmitted to the client using the traditional RFB protocol to ensure good quality. As for the high motion regions, they are sent to the client using the RTP protocol, aiming for low bandwidth consumption while maintaining decent video quality. On the client side, two corresponding receiving and decoding modules are set up to handle their respective tasks. The received data is decoded, merged with other images, and placed in the data buffer, awaiting display on the client side.

4. EXPERIMENT AND RESULT ANALYSIS

4.1. Test Environment

The devices include two computers, one serving as the server and the other as the client. The table 1 below displays the specific configurations of each computer.

Table 1. Configurations of each computer

	cpu	cores	main frequency	ram	gpu
server	AMD Ryzen 5 2600	6	3.4ghz	4g	GTX1060
client	Intel e3	4	3.2ghz	4g	GTX880M

During the experiment, the MSI Afterburner and RivaTuner Statistics Server programs were used to monitor various parameters of the server and client, with a particular focus on CPU usage. The main objective of this study is to enhance the capabilities of the newly designed optimized VNC remote desktop system in handling image processing tasks such as videos and games. This includes conducting basic performance testing of the system. The key performance indicators for the testing include the CPU usage and network bandwidth utilization of both the server and client during the operation of the VNC system, particularly during video playback.

In the experiment, three application scenarios were set up: The first scenario involved pure text processing tasks, such as working with Word documents. The second scenario focused on playing videos in a non-fullscreen mode, where the non-video portion still contained textual information. The third scenario involved playing videos in fullscreen mode. Finally, tests were conducted to measure the CPU usage and network bandwidth for each of these three scenarios.

4.2. Result and Analysis

A． Result

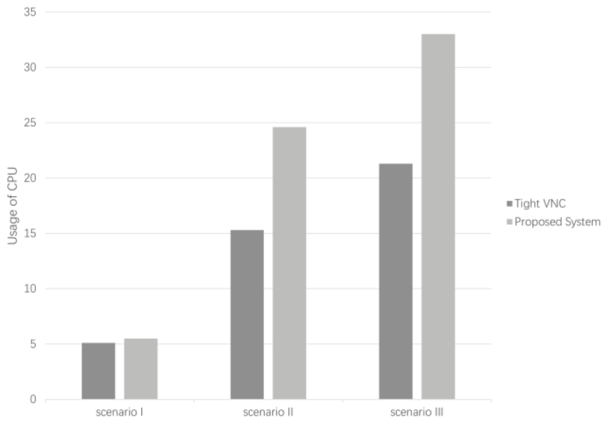

Figure 3 Usage of CPU in the server

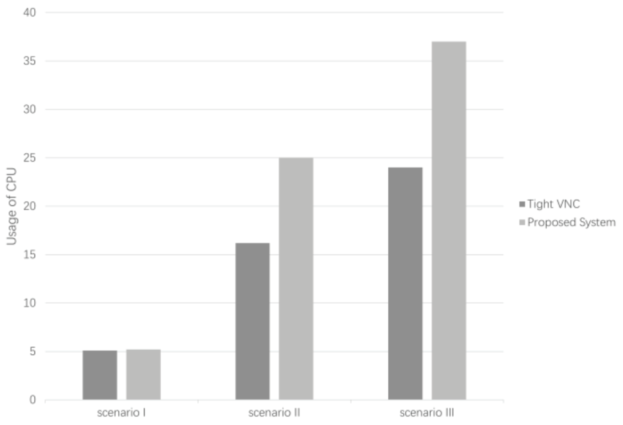

Figure 4 Usage of CPU in client

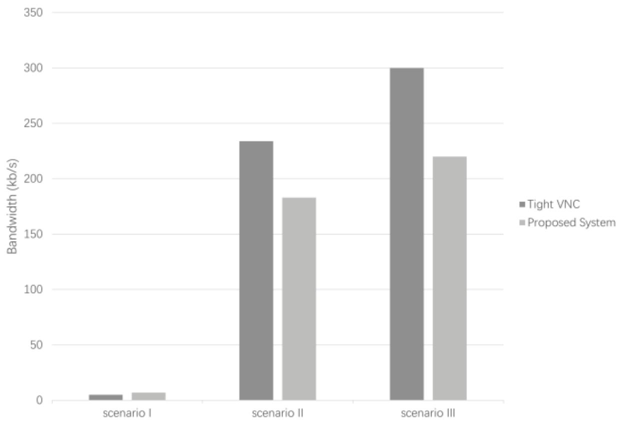

Figure 5 Bandwidth utilization

B． Analysis

Based on the results of the aforementioned experiment, analyzing Figure 3 reveals that as the complexity of the screen image increases, the CPU usage of any VNC system also increases. Moreover, in the proposed system, the CPU usage is 10% higher in subsequent scenarios. A similar trend can be observed on the client side, as shown in Figure 4. The proposed system consumes more CPU resources on the client side.

Through careful analysis of this phenomenon, we can easily identify the reasons behind it. In the proposed system, after receiving update requests, the server performs encoding and transmission within a predetermined time frame. Additionally, before encoding, the server needs to calculate the high and low motion regions, which increases the CPU resource usage on the server side. Consequently, the client continuously receives and decodes data, leading to an increase in CPU usage on the client side as well.

Although the CPU usage has increased, it is evident from Figure 5 that the proposed system has reduced bandwidth usage. This is because as the screen image becomes more complex and changes occur more rapidly, more areas are identified as high motion regions. The high motion regions are encoded and decoded using the high compression ratio H.264 algorithm. While this method consumes more CPU resources, it significantly reduces the amount of data transmitted, resulting in better bandwidth utilization compared to the Tight VNC system.

5. Conclusion

This paper proposes an optimised idea and method based on the disadvantages of the lazy update mechanism of the VNC system, the unfavorable encoding method of the video data, and the high bandwidth consumption of the transmission of highly variable images. For the lazy update mechanism, a parameter that is dynamically adjusted according to the network delay is used to control the minimum time for the client and server to communicate at one time and to continuously update the data during this period. For the VNC video transmission problem, a video frame high and low motion area identification algorithm is proposed. The algorithm distinguishes between high and low motion zones of the current frame by calculating the modal length of the motion vector. The high motion region is then compressed using the H.264 coding of the integrated FFmpeg codec, while the relatively low motion region is still compressed and transmitted using the RFB protocol. In the end, it was confirmed in the experiments that this method would increase the CPU usage of the system to a certain extent, both on the client side and on the server side. However, in terms of bandwidth usage, it also achieves a maximum 25% advantage over Tight VNC.

6. Acknowledgments

This work was supported in part by the Fundamental Research Funds for the Central Universities(Grant No. 2682022CX066）and the Science and Technology Research and Development Plan of China Railway(Grant No. N2022S006).

References

[1] C. Xu, D. Li, W. E. Wong, and M. Zhao, “Service caching strategy based on Edge Computing and Reinforcement Learning,” International Journal of Performability Engineering, vol. 18, no. 5, pp. 350–358, 2022.

[2] C. Miyachi. What is “Cloud”? it is time to update the NIST definition?[J]. IEEE Cloud Computing, 2018, (3): 6-11.

[3] J. Pateria, L. Ahuja, and S. Som, “Critical path to place decoys in Deception biota,” International Journal of Performability Engineering, vol. 18, no. 12, pp. 854–862, 2022.

[4] Zhu Yongqiang , Tang Xiong . Analysis and Research on VNC-based Remote Desktop Transfer Protocol [J]. Computer System Applications Computer System Applications,2016,25(11):284-287.

[5] Rao S S, Vin H M, Tarafdar A. Comparative evaluation of server-push and client-pull architectures for multimedia servers[J]. Proceedings of NOSSDAV’96, 1996: 45-48.

[6] Taylor C, Pasquale J. Improving video performance in VNC under high latency conditions[C]//2010 international symposium on collaborative technologies and systems. IEEE, 2010: 26-35.

[7] Wu Xiaoyu. Analysis of RFB protocol and video playback performance improvement in VNC system [D]. Nankai University, 2008.

[8] Fei-Die Liang , Jin-Tao Li , Hong-Zhou Shi . Coding methods in virtual network computing (VNC) protocols [J]. Computer Applications,2004,24(6):93-95.

[9] D. De Winter, P. Simeons, L. Deboosere, F. De Turck, J. Moreau, and B. Dhoedt, P. Demeester, “A Hybrid Thin -Client Protocol for Multimedia Streaming and Interactive Gaming Application” Proc. Int. Workshop on Network and sOperating System Support for Digital Audio and Video, Aug. 2006.

[10] Tan K J, Gong J W, Wu B T, et al. A remote thin client system for real time multimedia streaming over VNC[C]//2010 IEEE International Conference on Multimedia and Expo. IEEE, 2010: 992-997.

[11] Nguyen, Tien-Dung, Seungun Choe, and Eui-Nam Huh. "An efficient mobile thin client technology supporting multi-sessions remote control over VNC." 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE). Vol. 3. IEEE, 2012.

[12] Bross, Benjamin, et al. "Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc)." Proceedings of the IEEE 109.9 (2021): 1463-1493.

[13] J. -R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1669-1684, Dec. 2012, doi: 10.1109/TCSVT.2012.2221192.

[14] Bao, Wenbo, et al. "Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement." IEEE transactions on pattern analysis and machine intelligence 43.3 (2019): 933-948.