Design of Campus Network Streaming Media Live Broadcasting System

Yinqian Cheng[a], Xinge Liu[b], Lei Yu[c,*]

Information Network Center, China University of Geosciences (Beijing), Beijing, China
[a]chengyq@cugb.edu.cn, [b]1004132131@cugb.edu.cn, [c]yul@cugb.edu.cn
[*]Corresponding author

Abstract: *Currently, video live streaming has become a hot topic of research. Students, who have more free time, have become the main audience of online live streaming, but this has also put considerable pressure on the egress of campus networks. At the same time, due to the lack of cable TV signals in dormitories, students' demand for live TV programs cannot be met in real time. Considering the above factors, this paper takes China University of Geosciences (Beijing) as an example and combines the actual environment and usage needs of universities to analyze the actual topology of the campus intranet in detail. Targeted and precise adjustments are made to the traffic structure, and a campus intranet media streaming live system is designed. Currently, this system is running stably at China University of Geosciences (Beijing), providing strong live streaming support for various activities and major events, significantly improving the traffic structure of egress, saving a lot of traffic costs for faculty and students, and achieving significant optimization effects in terms of network quality and service diversity of the campus network.*

Keywords: *Stream media, Video, Live, Campus network*

1. Introduction

With the rapid development of Internet technology, the trend of constructing campus networks has quickly emerged in universities [1]. Although many universities have built their own campus networks, their current applications are not optimistic, and there are many problems that need to be solved [2]. Generally, the speed of campus networks is fast, providing certain conditions for the transmission and playback of audio and video on campus networks [3]. With the increasing popularity of the concept of online live streaming, the barrier to entry for live streaming has gradually lowered, and "watching live streams" has become a part of people's daily lives. However, this live streaming boom has brought enormous traffic expenses, resulting in excessive load on the egress of campus networks in universities, thereby affecting the network user experience. Traditional architectures of campus networks in universities often struggle to cope with the massive influx of traffic. On the one hand, faculty and students on campus need normal academic use of the network; on the other hand, the increasing demand for entertainment poses challenges to the campus network. The complex intranet architecture and chaotic egress structure of the campus network often result in a trade-off situation. Meanwhile, cable TV signals are usually not available in dormitories on campus, which leads to students seeking external live streaming on the intranet terminals driven by their demand for TV viewing. Especially during major social events of public concern, numerous intranet terminals repeatedly access the same live streaming content through the campus network gateway, resulting in redundant expenses.

In the past, when there was live broadcasting of major events, multiple internal network terminals would independently access the same content through the campus network's external gateway, resulting in redundant traffic expenses for the gateway[4]. The system designed in this paper can pull media content into the campus network through a dedicated external network route, and terminals can directly access the corresponding content from the internal network, thus saving external network gateway expenses. This system design can effectively optimize the traffic management of the campus network, reduce the burden on the external network gateway, improve the network usage experience, and meet the demands of students for live broadcasting and TV programs, especially in cases where TV signals are not provided in student dormitories. By sharing live broadcasting content within the internal network, dependence on the external network can be reduced, traffic expenses can be lowered, and the overall performance and user satisfaction of the campus network can be improved.

2. Introduction to System-Related Technologies

2.1. PHP

PHP is a popular general-purpose scripting language. PHP is widely applicable, highly scalable, and powerful in terms of performance. In the design of this system, PHP is responsible for functions such as user page rendering, data interaction, and internal cluster control. Due to the use of PHP 8.0 version in this system, it performs well in various tests, especially stress testing.

2.2. RPC

Remote Procedure Call (RPC) is a computer communication protocol that allows programs running on one computer to call subroutines on another computer without the need for additional programming for this interaction. In the cluster mode of this system, the remote calling process of each node is not implemented as a standard RPC instance, but rather through HTTP requests and callbacks to achieve the corresponding functionality of RPC. Strictly speaking, the remote calling in this design does not fully comply with the RPC standard, but because the final implementation achieves the same result as RPC, it can be considered as a type of RPC implementation in a broad sense.

2.3. Streaming Pull and Push

Streaming media, as a type of data stream, has certain differences in nature compared to regular static file content. The basic and fundamental operations for handling streaming media content are pulling and pushing, which are essential for ensuring smooth playback. There are various types of streaming media servers available in the market, including commercial ones such as Wowza, Flussonic Media Server, and Adobe Media Server, as well as open-source options such as gstreamill, Nimble Streamer, Simple RTMP Server, and Nginx RTMP Module.

2.4. Transcoding

There are various types of streaming media formats available, and to ensure a convenient and seamless viewing experience for ordinary users, it is necessary to minimize the effort required on the client side. One effective approach is to transcode all pushed content into a consistent encoding format that can be directly played on clients. Common open-source transcoding solutions include FFmpeg, Libav, x264, and others. This simplifies the process, makes it fast and efficient, and ensures that all streaming media on the system can be played smoothly on all clients.

2.5. Distribution

Distribution refers to the process of delivering the final processed media content to the clients (viewers). Due to the diverse requirements of different types of media, there are various distribution methods available, each with its own focus. Commonly used distribution methods include RTMP (Real-Time Messaging Protocol), RTSP (Real-Time Streaming Protocol), HTTP-FLV (HTTP-based Flash Video), and HLS (HTTP Live Streaming).

3. System Design

In the development process of this system, commercial media server software was initially excluded as the hosting server. Among all the open-source media server software options, Nginx RTMP Module was ultimately chosen as the main hosting core[5]. As a module of Nginx, it can seamlessly integrate with various callback operations of Nginx, providing a robust and flexible solution.

3.1. Core Scheduling System

As the core scheduling system of the cluster, it is responsible for receiving status information from all other servers and issuing control commands to other servers based on factors such as the number of viewers, server load, and network status. It also serves as the response server for direct access to the main domain name by users. During system deployment, simply pointing the domain name to the core scheduling system ensures that users and other nodes can access the relevant resources of the core scheduling system correctly. Even if the IP address of the core scheduling system needs to be changed due to various reasons, the system will provide feedback on its operational status to the core scheduling system as soon as it is restored.

3.2. Streaming Tanscoding Node

After pulling the media stream, if it cannot be directly processed through HLS segmenting, it will be converted by the transcoding node. In this node, PHP is used to control FFmpeg to process external input streams and push the output streams to designated locations. In the actual production environment, this system is deployed on 2 servers as transcoding nodes, with one server serving as a backup that only handles workload when the primary transcoding node is overloaded or unavailable.

3.3. Streaming Pull Node

The pulling nodes are responsible for obtaining the source media streams from the internet or through user-initiated pushes, and reporting the acquired formats to the core scheduling system. After evaluation, the media streams are pushed to designated servers. The core scheduling system can start a live stream channel through scheduled tasks or manual activation. In the actual system deployment, 2 pulling nodes are set up. One node serves as the main pulling node for educational TV channels from the external IPv6 network, using a fully fiber-optic network that can handle concurrent tasks of pulling streams from hundreds of channels. The other node serves as a backup pulling server for external IPv4 public network media sources, working as an emergency backup pulling server in case of special requirements or abnormal IPv6 signals.

3.4. Edge Distribution Node

The system design deploys edge distribution nodes in concentrated user areas such as educational zones, dormitory areas, office areas, etc., to achieve the goal of distributing media content to users' local areas. This edge distribution node architecture can improve the user's playback experience and reduce the load on the backbone network. When a user opens the playback page of a corresponding channel, the core scheduling system returns the actual address of the corresponding edge distribution node, guiding the user's browser to directly obtain the actual playback content from that node. This avoids long-distance transmission of media content from the core server to user terminals, reduces latency, improves loading speed, and enhances the user's viewing experience. At the same time, by deploying edge distribution nodes at multiple locations, the system can achieve efficient distribution of media content, reducing the load on the backbone network. This distributed architecture helps improve system stability and fault tolerance, as even if a node experiences an exception or failure, other nodes can continue to provide services, ensuring system reliability.

3.5. Client Design

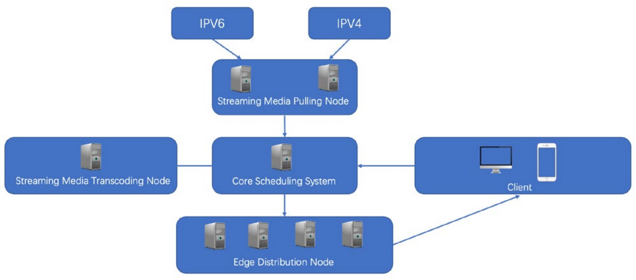
IPV6
IPV4
Streaming Media Pulling Node
Streaming Media Transcoding Node
Core Scheduling System
Client
Edge Distribution Node

Figure 1: System Architecture

In the client-side design, a cross-platform adaptive approach is adopted to reduce redundant development efforts. The Clappr open-source project is introduced to implement the playback functionality on both desktop and mobile platforms, with the functional code shared between them to achieve code reusability[6]. Users only need to access the main domain pointing to the core scheduling system to obtain channel configuration information and watch media content. This design approach simplifies the development process of the client, reduces maintenance costs, and ensures consistency in user experience between desktop and mobile platforms. In the desktop environment, to address the issue of some browsers lacking built-in support for HLS (HTTP Live Streaming), the system also introduces hls.js as an additional decoding feature. hls.js is an open-source JavaScript library used to implement HLS playback functionality in browsers that do not support HLS natively. By introducing hls.js, the system can achieve smooth playback of HLS format media content on desktop browsers, ensuring a seamless playback experience for desktop users.

After completing the aforementioned design, the architecture of the system is shown in Figure 1.

3.6. Workflow

The core scheduling node checks the database every three minutes to obtain the streaming media information to be played based on the channel information, external source addresses, and the schedule stored in the database. When selecting playback nodes, the core scheduling node uses the clCluster::getStatus method to consider the load status of the pulling and transcoding nodes, the physical location of the distribution nodes, and potential high-traffic audience concentration areas to choose the most suitable node combination. Once the selection is made, the core scheduling node calls the clRPC::callRemote Command method to send task-specific configurations to the selected nodes. Upon receiving the request, the pulling node checks the format of the source stream. If the source stream format is compatible with the client browser for direct playback, the pulling node directly pushes the fetched streaming media to the specified distribution node. If the source stream format is incompatible, the pulling node pushes the streaming media to the transcoding node and uses FFmpeg to perform format conversion. The transcoding node then pushes the transcoded streaming media to the distribution node.

Once the client browser opens the playback page, it requests channel information, including the streaming address of the actual distribution node, from the core scheduling node. Once the streaming address is obtained, the client browser directly retrieves the actual streaming content from the distribution node and begins playback.

This design approach enables the system to select the most appropriate node for media distribution based on factors such as node load and location. Additionally, the introduction of transcoding nodes allows for the handling of different media formats, ensuring playback compatibility. By requesting the streaming address from the core scheduling node, the client browser can retrieve content from the nearest distribution node directly, thereby improving playback efficiency and user experience.

4. Conclusion

The streaming media live broadcasting system implemented at China University of Geosciences (Beijing) has successfully alleviated the pressure on the campus network's external Internet exit and increased the usage of the IPv6 network. It has provided a comprehensive channel for watching cable TV channels for students on campus, effectively solving the issue of cable TV signal supply in dormitories. During the system design process, challenges such as content distribution and access control for both IPv6 and IPv4 networks were successfully addressed, and a robust adaptive solution for wired and wireless networks was provided, ensuring a consistent streaming media viewing experience across desktop and mobile devices. This system presents a new solution for campus intranet live broadcasting of campus events in higher education institutions, overcoming the issues of traffic cost and external Internet exit pressure associated with using external live broadcasting platforms for campus events with large audiences. It provides a powerful solution for promoting live broadcasting of future campus events.

References

[1] Y. Fan, Y. Wang. The study of part-peer-based streaming service system [J]. Electronic Design Engineering, 2017, 0(4):154-157.
[2] X. N. Wang, J. G. Han. Research on IPv4/IPv6 Transition Technology of Network Convergence in Smart Campus Construction [J]. Journal of Xianyang Normal University, 2022, 37(6):23-27.
[3] J. H. Sun, P. Yang. Construction and Application of Video Resource Management System Based on Campus Network [J]. Electronic Components and Information Technology, 2021, 5(05):28-29.
[4] M. M. Xiao, J. C. Liu, Y. L. Li, Y. Ma. Exclusive Cloud Architecture Design and Implementation for Live Streaming Platforms [J]. The Chinese Journal of ICT in Education, 2021(3):88-91.
[5] Arut. Nginx-rtmp-module wiki[EB/OL]. https://github.com/arut/nginx-rtmp-module/wiki. Accessed on 2023-04-20.
[6] Clappr.io. Clappr wiki[EB/OL]. https://github.com/clappr/clappr/wiki. Accessed on 2023-04-20.