UDC 004.55
DOI: 10.15587/1729-4061.2025.339985

DEVELOPMENT OF ADAPTIVE CONGESTION CONTROL MECHANISM FOR REAL-TIME MULTIMEDIA STREAMING IN VARIABLE NETWORK CONDITION

This research focuses on real-time multimedia streaming using RTP and RTCP protocols. The main issue addressed is that standard RTP/RTCP congestion control is inadequately adapted to changing and unstable network conditions, resulting in increased packet loss, end-to-end latency, unstable bitrates, and poor video quality. A dynamic bandwidth-adaptive congestion control mechanism was developed for RTP streaming, which utilizes RTCP feedback to dynamically change the bitrate and framerate in real time during the streaming session. Controlled experiment results show that average packet loss decreases from 8.2% to 3.4%; end-to-end latency decreases from an average of 220 ms to 135 ms; and provides a more stable average bitrate than standard RTP/RTCP systems. Furthermore, this system also provides a more stable average framerate than standard RTP/RTCP systems and a higher average framerate under poor network conditions. This result can be attributed to the ability of the adaptive mechanism to continuously monitor packet loss, interference, and delays in addition to reacting immediately to conditions instead of waiting for RTCP reports to appear at fixed time intervals. A key point regarding the proposed design is the integration of bitrate and framerate to ensure smooth playback and user enjoyment with reduced risk of interruption and improved stability in dynamic and unpredictable network environments. This contribution can be practically applied in real-time applications, such as video conferencing, telemedicine, or live streaming while traversing mobile or wireless networks where conditions are always dynamic and unpredictable. The proposed method can be practically applied under unfavorable internet network conditions, which is an advantage of this method

Keywords: adaptive system, multimedia, network congestion, real-time transport control protocol, video streaming

Marvin Chandra Wijaya
Philosophy Doctor of Information Communication Technology, Associate Professor
Department of Computer Engineering
Maranatha Christian University
Suria Sumantri str., 65,
Bandung, Indonesia, 40164
E-mail: marvin.cw@eng.maranatha.edu

Received 27.06.2025
Received in revised form 27.08.2025
Accepted date 08.09.2025
Published date 28.10.2025

How to Cite: *Wijaya, M. C. (2025). Development of adaptive congestion control mechanism for real-time multimedia streaming in variable network condition. Eastern-European Journal of Enterprise Technologies, 5 (9 (137)), 54–63. https://doi.org/10.15587/1729-4061.2025.339985*

1. Introduction

Multimedia streaming delivers audio, video, and other media from a server over the internet to users in real time [1]. Multimedia streaming is a ubiquitous part of everyday life today [2]. It is widely used for various purposes, such as video calls, online learning, entertainment platforms, live events, and gaming. Multimedia streaming is becoming increasingly popular due to advances in internet speed and bandwidth. However, unstable internet conditions make real-time multimedia streaming vulnerable to network conditions [3]. Problems, including packet loss, delays, jitter, and bandwidth fluctuations, are still common [4]. These problems are more prevalent on cellular and wireless networks. These issues directly impact the user experience, causing video lag, poor audio quality, and interference. Stable transmission and adaptive system mechanisms are crucial for various applications such as telemedicine, remote work, virtual classrooms, and emergency communication systems [5].

Fast and stable communication over a network requires a system that utilizes the real-time transport protocol (RTP) and the RTP control protocol (RTCP). Media such as text, video, audio, animation, and many others use RTP as their data delivery protocol [6]. RTCP is used to monitor the quality of an application's connection while streaming data [7]. A multimedia presentation, consisting of video and audio, is combined with a shared screen or text chat and then sent over the internet [8]. The data is often compressed using codecs such as H.264 for video or Opus for audio to allow faster transmission over the network [9].

One of the biggest constraints in multimedia streaming is network congestion. Congestion occurs when too much data is sent over the network simultaneously. Congestion is unavoidable and uncontrollable by the system, as users control much of the data being transmitted. When the network is overloaded, data will be lost or the speed will slow down, decreasing streaming quality. Congestion is more likely to occur on wireless networks such as 4G, 5G, or Wi-Fi, where signal strength and interference can fluctuate rapidly [10]. Public network usage can cause bottlenecks for all network users. Congestion bottlenecks often occur in crowded areas or during peak hours. While RTP/RTCP protocols provide basic feedback mechanisms, they have limitations when dealing with today's highly variable and congested networks. In particular, RTCP feedback is often too slow to respond to rapid changes in network conditions, leading to persistent quality degradation.

Therefore, the scientific topic of developing adaptive congestion control mechanisms for RTP/RTCP remains important and crucial. A system that can adjust in real time based on network conditions can significantly improve the quality of multimedia streaming. A system that responds intelligently and adaptively to network changes can help reduce delays, prevent video interruptions, and provide users with a better multimedia streaming experience.

2. Literature review and problem statement

It has been shown in [11] that adapting the transmission rate can effectively mitigate packet loss and improve real-time performance in multimedia streaming. However, some issues are still unresolved, such as how to quickly adjust to abrupt bandwidth changes, demand less computation from resource-limited devices, and respond quickly to short-term variations in the network. These problems are often due to the challenges of predicting network behavior, the cost of fast processing, and the time taken for feedback to be returned through the network.

A comparative analysis of adaptive congestion control algorithms used in RTP-based streaming evaluated the advantages and disadvantages of several frequently used in practice. The outcome in [12] is that while many adaptive congestion control algorithms, such as GCC. NADA's rate adjustment will adaptively calculate more quickly, but it shows issues of fairly sharing resources with others. Others, like SCReAM may limit the build-up of data in the queue, but will allow portions of bandwidth to go unused. All of these problems could be attributed to design limitations that consistently seek to reconcile fairness, responsiveness, and capacity. There may be opportunities for a hybrid approach that combines positive features of these different systems. There is justification in pursuing a balance of the consistency space in a more systematic way for adaptive congestion control in RTP-based streaming of real-time multimedia.

Current congestion control mechanisms have difficulty achieving a balance between low latency, high throughput, good adaptability, and fairness, mainly due to the limitations of available control strategies and the constraints of the convergence objective. The approach in [13] was proposed from the observation of a linear relationship between RTT variances. The method contributes to fairness and low latency in the delivery rate to the same queue load. Small packets (8 packets in their experiments) are generated and maintained in an unstable network. Latency is treated as a performance measure to be improved. The system modulates adaptability to react to changing network conditions. However, based on simulations, the system's speed in balancing fairness, responsiveness, and stability still needs to be improved.

It has been shown that congestion management in smart grid networks remains a challenging issue, especially when the networks use unreliable protocols such as UDP. A major unresolved issue is their ability to cope with data management in constantly changing urban networks. This is mainly due to TCP's inability to cope with congestion in streaming networks. The high cost of designing adaptive algorithms for complex networks also poses a problem in improving the quality of multimedia streaming. The most important solution to all these complex issues is the implementation of reinforcement learning (RL) and deep Q-neural networks (DQN) that can be trained through interaction with the network. The subject area [14] is being implemented in a recent study, which tested modified RL and DQN algorithms in Montreal, Berlin, and Beijing. The results show that modified RL and DQN algorithms result in substantial improvements in packet delivery, network throughput, fairness between traffic sources, packet delay, and a wide range of quality of service. However, adaptive congestion control with self-learning algorithms needs further enhancement.

Congestion control in multimedia streaming remains a challenging problem. This problem occurs due to the unpredictable nature of network traffic, the difficulty of running real-time algorithms on lightweight devices, and the cost of implementation. The development of better feedback and adaptive mechanisms capable of predicting congestion can be beneficial for improving the quality of multimedia streaming. Study [15] shows that the standard RTCP feedback is slow, fixed in a certain interval, and cannot predict future congestion. All this leads to the conclusion that efficient and adaptive congestion control for interactive multimedia streaming is a subject worthy of study. However, several major challenges remain, including fast and accurate adaptation to changing network conditions.

These unresolved issues are due to objective constraints (such as packet feedback rate and codec adaptation rate), the need for appropriate complexity design in multimedia programming algorithms, and the lack of integration with existing RTP infrastructure. Highly reliable packet communication [16] is a challenge for critical applications in future wireless networks.

Achieving highly reliable communication with an effective probability requires a new communication paradigm. Because monitoring equipment in mobile areas requires greater attention, monitoring the transmitted data is necessary. The use of neural networks, as in the study [17] can be used for track recognition from video streaming.

It has been shown that ensuring information security in real-time video streaming remains a challenge due to threats such as eavesdropping, data manipulation, and hacking. A persistent challenge is combining strong end-to-end encryption with low latency. Furthermore, the encryption must support modern codecs. A recent study [18] extended the uvgRTP transport library with Secure RTP and Zimmermann RTP, enabling encrypted 8K video streaming at high frame rates. However, this process has drawbacks, such as the need for a large internet bandwidth.

It has been identified that underwater multimedia transmission has major challenges such as narrow bandwidth, interference, and image distortion. The challenge is that standard RTP/RTCP congestion control is inadequately adapted to changing and unstable network conditions, resulting in increased packet loss, end-to-end latency, unstable bitrates, and poor video quality. To solve this issue, it is proposed a way to piece together autonomous RTSP transmission, RTP packet encapsulation, and RTCP feedback to control congestion. A recent study [19] designed a real-time RTSP transmission system for underwater panoramic cameras, which showed stable throughput, low packet loss below 0.5%, and effective real-time video delivery. However, this method can only be used for relatively short distances. Therefore, further studies are needed to determine stable multimedia streaming delivery over longer distances.

A study of adaptive streaming systems [20] shows that scalable video coding (SVC) can effectively adapt to bandwidth variations. By estimating available throughput using metrics such as packet loss, jitter, and data reception time, adaptive streaming systems can improve overall video quality as link capacity changes in the network. However, challenges remain, such as slow system adaptation after sudden bandwidth changes, challenges for lightweight devices due to increased complexity, and the need for testing under various real-world network conditions. Therefore, there is scope for further research in the area of congestion control schemes capable of addressing video quality and stability under challenging and variable network conditions.

3. The aim and objectives of the study

This study aims to improve the quality of real-time multimedia streaming using RTP and RTCP. This improvement is achieved through the design of an adaptive congestion control system that can respond to varying network conditions.

To achieve this aim, the following objectives are accomplished:

– to test the limitations of existing RTP and RTCP congestion control mechanisms, especially under unstable or variable network conditions;

– to design and implement an adaptive real-time tuning system by leveraging RTCP feedback to dynamically modify streaming parameters and evaluate video streaming quality under various network scenarios;

– to compare the results of the proposed system with standard RTP and RTCP systems.

4. Materials and methods

4. 1. Object and hypothesis of the study

Real-time multimedia streaming using RTP/RTCP experiences challenges according to the dynamic quality of the network. The standard RTP/RTCP feedback loop is periodic and slow, thus not responding swiftly to sudden changes in bandwidth, loss, delay, and jitter. Consequently, streams experience high packet loss, high latencies, unstable bitrates, and observable quality degeneration. The object of study is a real-time multimedia streaming system based on RTP/RTCP, which focuses on handling network congestion to maintain media quality in unstable network conditions.

The first problem is that standard RTP and RTCP congestion control mechanisms cannot effectively handle unstable or fluctuating network conditions, resulting in packet loss, latency, jitter, and unstable bitrates. The second problem is that current RTP/RTCP systems lack adaptive mechanisms to dynamically adjust bitrates and frame rates in real-time based on RTCP feedback, resulting in poor streaming quality when network conditions fluctuate. The third problem is that existing research lacks a systematic comparison between adaptive mechanisms and standard RTP/RTCP systems, so it is unclear how much improvement adaptive methods provide in practical scenarios.

This study hypothesizes that implementing an adaptive congestion control mechanism based on real-time RTCP feedback will improve the quality, stability, and responsiveness of multimedia streaming. The quality of RTP/RTCP multimedia streaming with adaptive congestion control will be better than that of a standard RTP/RTCP system.

Assumptions made in the study are that the network conditions simulated with Linux Traffic Control reasonably mimic the types of variance likely to be encountered in the real world, including packet loss, jitter, and bandwidth. Other assumptions are that RTCP feedback delays remain consistent across all trials and that the end devices have sufficient processing power to implement the adaptation logic in real time without introducing noticeable delays.

Simplifications adopted in the study are the use of only one video codec (H.264) to maximize consistency in the evaluation. Another simplification is the selection of only one type of video content to avoid variability due to complexity, instead of using content with varying motion or detail within the scene. A further simplification is that all experiments were conducted within a controlled laboratory network using a virtual LAN, rather than over a heterogeneous or wide area network. This allowed the researchers to observe the effects of adaptive congestion control on transport and video experiences, but it may limit how broadly the results can be applied to real-world scenarios.

4. 2. Evaluating the limitations of multimedia streaming congestion control

The first problem is that standard RTP and RTCP congestion control mechanisms cannot effectively handle unstable or fluctuating network conditions, resulting in packet loss, latency, jitter, and unstable bitrates. Under network congestion, standard RTP/RTCP will lose packets and become too slow, resulting in unreliable video quality, high latency, and obstruction. An experimental design was created to assess this issue and compare standard RTP/RTCP with the proposed adaptive solution.

This study will conduct a detailed analysis of how RTP and RTCP operate in real-time streaming to identify the limitations of the RTP and RTCP standards. RTP and RTCP are complementary standard protocols for real-time data transmission over IP networks, such as audio and video. The relationship between RTP and RTCP is based on how RTCP sends feedback and examines how RTP responds to this feedback during network congestion.

Therefore, this study will utilize simulation tools to observe the behavior of RTP/RTCP in real-time conditions. GStreamer software will be used to simulate multimedia streaming under various network conditions. Key performance indicators such as packet loss, jitter, delay, and video quality will be monitored and recorded.

Some experimental scenarios include the following:

– sudden bandwidth drops or increases;

– network congestion due to background traffic;

– variable delays and jitter, as found in cellular or wireless networks.

The experiment was conducted on a laboratory network of one sender and receiver connected via an emulated channel. The system was implemented on Linux using a GStreamer pipeline to transmit video over RTP/UDP. The transmitted video was encoded with H.264 encoding. A Python script was used to observe RTCP feedback and implement the adaptation logic in the proposed system. Network issues, including packet loss, bandwidth degradation, jitter, and latency, were all addressed through Linux Traffic Control. Solution using a Python script:

a) initialize the test environment:

– apply the network conditions defined by scenario;

– start a packet capture tool;

b) run the sender:

– construct and execute the GStreamer pipeline for the sender;

– GStreamer_pipelinesender ← filesrc → demux → decode;

– stream the video from Vsrc via RTP to the receiver;

c) run the receiver:

– construct and execute the GStreamer pipeline for the receiver;

– GStreamer_pipelinereceiver ← udpsrc → rtpbin → rtph264depay → avdec_h264;

– listen for incoming RTP and RTCP packets from the sender;

d) measure and record data:

– for Tdur seconds, monitor and record data on the receiver host;

– use RTCP feedback to measure Dloss and Djitter in real-time;

– analyze the packet data from the capture tool to obtain Ddelay and verify Dloss;

– calculate Dqos by comparing the received stream to the original stream;

e) end the scenario:

– stop the sender and receiver pipelines;

– stop the packet capture;

– remove the applied network conditions.

By observing the data collected in these various situations, this study can identify specific weaknesses in each condition. The data collected will be compared across various scenarios to analyze which issues are serious, fatal, and frequently occurring.

The solution to the problem of standard RTP and RTCP congestion control mechanisms being inadequate for variable or unstable network scenarios proved to be an experimental design. This study used the GStreamer simulation tool to observe the behavior of RTP/RTCP when experimenting with various network scenarios such as unexpected bandwidth drops, congested background traffic, varying delays, and varying jitter. The study tracked various performance metrics, such as packet loss, jitter, delay, and video quality, to determine specific weaknesses in the RTP/RTCP standards.

4. 3. Design and implementation of the adaptive RTP/RTCP mechanism

The second problem is that current RTP/RTCP systems lack adaptive mechanisms to dynamically adjust bitrates and frame rates in real-time based on RTCP feedback, resulting in poor streaming quality when network conditions fluctuate. Because RTCP reports occur at fixed intervals, when action is required due to sudden changes in network status, those changes are typically not implemented until it is too late, resulting in poor streaming quality. An adaptive congestion control system that responds immediately to RTCP feedback and recommends transmission parameter adjustments based on the observed RTCP values has been created.

A stepwise approach was used to design and implement an adaptive RTP/RTCP streaming system capable of customizing streaming parameters based on changing network conditions. This approach involved system architecture, implementation with GStreamer, modeling various network conditions, and experimental testing using the aforementioned metrics.

RTCP allows the receiver to periodically send reports, which in this case can include reports that provide packet loss rates, round-trip delay (RTT), and jitter. Real-time data from RTCP is then processed using Python to adjust RTP parameters. The system uses a real-time feedback loop based on the processed RTCP reports to determine the bitrate or frame rate of the RTP (Real-Time Transport Protocol) video stream. The decision logic is defined with several predefined thresholds. For example, if packet loss exceeds 5% or if the RTT exceeds 150 ms, the control module will reduce the streaming bitrate and frame rate to allow the stream to continue. Similarly, if network conditions improve, the system will gradually increase the quality settings.

This adaptive system is a Python-based control module added to the standard RTCP implementation within the GStreamer pipeline. It has continuous access to RTCP reports and aims to analyze key RTCP report values such as packet loss rate, jitter, and round-trip time (RTT). The adaptive control also monitors the current transmission parameters (bitrate and frame rate) and takes action to vary these parameters dynamically in conjunction with the RTCP analysis. For example, suppose packet loss exceeds 5% and a drastic increase in the observed RTT value occurs. In that case, the adaptive control system will reduce the bitrate by 100-200 kbps (and sometimes reduce the frame rate), then wait a period of time, re-evaluate the packet loss and RTT, and make additional changes. When observing a sustained increase in these RTCP parameters (packet loss less than 5%, constant, and reduction in the average RTT), the adaptive control system will gradually increase the bitrate while potentially increasing the frame rate in the hope of optimizing the quality of video service without causing instability at the network transport layer. The proposed mechanism will include a decision engine that works as follows:

– RTCP feedback is received periodically (every 1–2 seconds);

– the system analyzes key metrics;

– reduce the bitrate and frame rate if packet loss exceeds 5%;

– increase buffering or reduce the packet rate if jitter is greater than 30 ms;

– reduce the packet transmission frequency if RTT is greater than 150 ms.

Algorithm 1 will be implemented using a multimedia framework such as GStreamer. A script with this logic is added in Python to control the bitrate and frame rate dynamically. RTCP data will be recorded using the built-in monitoring tool.

Algorithm 1: Adaptive RTP Congestion Control Based on RTCP Feedback

```
Begin
    Bitrate ← 1500   // In Kbps
    Frame_Rate ← 30  // In Fps
    Min_Bitrate ← 500
    Max_Bitrate ← 2000
    Min_Frame_Rate ← 15
    Max_Frame_Rate ← 30
    Start_Stream()
    While Streaming_Is_Active Do
        Rtcp_Feedback ← Get_Rtcp_Report()
        Packet_Loss ← Rtcp_Feedback.Packet_Loss
        Jitter ← Rtcp_Feedback.Jitter
        Rtt ← Rtcp_Feedback.Round_Trip_Time
        If Packet_Loss > 5 Or Rtt > 150 Then
            Bitrate ← Max(Bitrate * 0.8, Min_Bitrate)
            Frame_Rate ← Max(Frame_Rate - 5, Min_Frame_Rate)
        Else If Jitter > 30 Then
            Bitrate ← Max(Bitrate * 0.9, Min_Bitrate)
        Else If Packet_Loss < 1 And Jitter < 10 And Rtt < 80 Then
            Bitrate ← Min(Bitrate * 1.1, Max_Bitrate)
            Frame_Rate ← Min(Frame_Rate + 2, Max_Frame_Rate)
        End If
        Update_Stream_Settings(Bitrate, Frame_Rate)
        Wait(2 Seconds)
    End While
    Stop_Stream()
End
```

Due to its flexibility and real-time processing capabilities, this adaptive streaming system prototype was built using the

GStreamer multimedia framework. The system consists of the following streaming pipelines:

– sender side: a GStreamer pipeline that captures and encodes the video source, sends it over RTP, and incorporates RTCP feedback handling;

– receiver side: a related pipeline that decodes and plays the video and generates RTCP reports for the sender.

A GStreamer pipeline, shown in Fig. 1, captures and creates a video stream with RTP and RTCP payloads. The adaptive logic was developed using a Python script layered on top of the standard GStreamer pipeline (Algorithm 2). The Python script acts as an RTCP listener, receiving all RTCP messages. The data is then processed, and the results are applied to dynamic commands in the encoder pipeline (to reduce or configure the bitrate or frames per second).

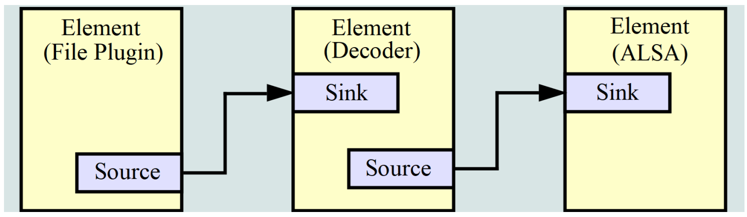

Fig. 1. Gstreamer pipeline

Algorithm 2: Real-Time Adaptive GStreamer Script (Python)

```
import gi
gi.require_version('Gst', '1.0')
gi.require_version('GstRtp', '1.0')
from gi.repository import Gst, GObject
import time
Gst.init(None)
pipeline = Gst.parse_launch("""
    filesrc location=sample.mp4 ! decodebin name=dec
    dec. ! videoconvert ! x264enc name=encoder tune=zerolatency bitrate=1500 speed-preset=superfast ! rtph264pay !
    queue ! udpsink host=127.0.0.1 port=5000 """)
encoder = pipeline.get_by_name("encoder")
def monitor_rtcp_and_adapt():
    bitrate = 1500
    while True:
            packet_loss = get_packet_loss_simulated()
            jitter = get_jitter_simulated()
            rtt = get_rtt_simulated()
            if packet_loss > 5 or rtt > 150:
                    bitrate = max(int(bitrate * 0.8), 500)
            elif jitter > 30:
                    bitrate = max(int(bitrate * 0.9), 500)
            elif packet_loss < 1 and jitter < 10 and rtt < 80:
                    bitrate = min(int(bitrate * 1.1), 2000)
            encoder.set_property("bitrate", bitrate)
            print(f"Adjusted bitrate to: {bitrate} kbps")
            time.sleep(2)   # interval between RTCP checks
def get_packet_loss_simulated():
    return random.choice([0.5, 2, 6, 8])
def get_jitter_simulated():
    return random.choice([5, 10, 20, 35, 50])
def get_rtt_simulated():
    return random.choice([50, 100, 160, 200])
pipeline.set_state(Gst.State.PLAYING)
import threading
import random
threading.Thread(target=monitor_rtcp_and_adapt, daemon=True).start()
loop = GObject.MainLoop()
try:
    loop.run()
except KeyboardInterrupt:
    pass
pipeline.set_state(Gst.State.NULL)
```

The planned architecture is composed of two planes, a media plane which transmits H.264 video over the RTP/UDP protocol and a control plane processing RTCP reports to provide live updates of the encoder. The two planes are maintained in a single GStreamer pipeline on an off-the-shelf Linux machine, implementing control logic in Python to adjust encoder properties during execution. The software stack is GStreamer 1.x plugin preconfigured with low-latency baseline profile. If packet loss was greater than 5% from control modules reporting, it would cause reduction of both bitrate and frame rate. If RTT was greater than 150 ms, it would also cause reduction in bitrate and frame rate. If jitter of packets was greater than 30 ms, it would cause a subsequent reduction of encoder bitrate. As conditions improved (packet loss of < 1%, jitter of < 10 ms, RTT of < 80 ms), bitrate and frame rate would be incrementally increased. Dead-bands and hold times were also used to mitigate oscillations.

The solution to addressing network congestion is to create and implement an adaptive congestion control system. This control system will use a Python-based control module, which will be added to the standard GStreamer pipeline. The control logic will run in a separate thread/event loop, dynamically adjusting the bitrate and frame rate of the video stream. The control logic will check for well-defined thresholds; for example, if packet loss exceeds 5%, or the RTT is greater than 150 ms, the system will reduce the bitrate and frame rate.

4. 4. Methodology for comparative performance evaluation

The third problem is that existing research lacks a systematic comparison between adaptive mechanisms and standard RTP/RTCP systems, so it is unclear how much improvement adaptive methods provide in practical scenarios. Without a controlled evaluation method, there is no way to determine whether there are differences between the two systems reliably. The adaptive and the baseline RTP/RTCP systems were evaluated fairly using identical test conditions – the same video source, codec (H.264), hardware environment, and network scenarios. The identical network scenarios included periods of normal conditions, quality degradation (packet loss, jitter, and bandwidth reduction), and recovery. Each scenario lasted 60–70 seconds, with performance data aggregated over 5-second periods.

The success of the proposed adaptive RTP/RTCP streaming system was measured using an experimental approach, comparing the two proposed systems. The experiment was structured by creating two streaming systems operating in parallel. The first system was designed with a standard RTP/RTCP configuration, and the second system used an adaptive RTP/RTCP configuration. These systems were built using the same approach and tested under identical network conditions. The performance of each streaming system was measured using controlled and repeatable experiments. The first streaming system used a standard RTP protocol configuration with a fixed bitrate and frame rate. The second streaming system used an adaptive system equipped with an algorithm capable of continuously monitoring RTCP feedback (packet loss, jitter, and round-trip time). The results of this feedback were used to adjust the video bitrate and frame rate precisely, simultaneously, and linearly. Both systems were designed to use the same video source and encode it in the same settings. Each video stream used H.264 encoding for testing.

However, the data and evaluation metrics included packet loss rate, bitrate stability, round-trip time (RTT), jitter, frame rate, and video quality, which was evaluated using PSNR values. Data was recorded continuously and evaluated, with average, minimum, and maximum values reported. All scenarios were repeated three times for both systems, ensuring reproducibility.

Several different and unstable network conditions were simulated using traffic control (TC) to simulate real-world network conditions. Traffic control was designed with network degradation and recovery scenarios at a controlled rate. The network condition changes implemented were as follows:

– packet loss (ranging from 2 to 10%);
– bandwidth throttling (from a maximum bandwidth of 1500 kbps to a standard 400 kbps);
– artificial delay or latency (up to 300 milliseconds);
– jitter (or delay variation, between 10 ms and 60 ms).

The test scenario lasted between 60 and 70 seconds, covering the network moving through three phases: a normal phase (with stable parameters), a degraded phase (with applied interference), and a recovery phase (where degraded parameters are restored over time). Both systems were tested independently using the same network scenario. Each experiment was repeated multiple times to avoid random chance influencing the results.

The solution to the lack of systematic comparison between adaptive and standard systems is a controlled and repeatable experimental methodology. This methodology involves measuring the proposed adaptive system and a baseline standard RTP/RTCP system under identical test conditions and network conditions. Network conditions include intervals of normality, quality degradation (i.e., packet loss, jitter, and bandwidth degradation), and a return to normality. The experiments are repeated multiple times to explain the results and to ensure their reliability and reproducibility.

5. Research results of the adaptive congestion control system for RTP/RTCP in real-time multimedia streaming

5. 1. Experimental data on limitations of real-time transport congestion control

Data acquisition was performed to analyze how RTP (real-time transport protocol) and RTCP (real-time transport control protocol) operate in real-time streaming and identify their limitations using Wireshark and GStreamer. This process involved recording network traffic during a live streaming session to observe the actual behavior of both protocols:

– tool: GStreamer + Wireshark;
– stream type: RTP video stream (H.264 codec);
– test duration: 2 minutes per scenario;
– feedback interval (RTCP): 5 seconds;
– resolution: 720p @ 30fps.

Network conditions:

– scenario A: stable network (no congestion);
– scenario B: sudden bandwidth drop (from 5 Mbps to 1 Mbps);
– scenario C: random jitter and packet loss (mobile/wireless simulation).

To evaluate the performance of the proposed mechanisms, key metrics were collected across different scenarios. The results of these measurements are summarized in Table 1 for easy comparison.

Table 1

Metric data from various scenarios

Metric	Scenario A (stable)	Scenario B (bandwidth drop)	Scenario C (jitter + loss)
Avg. packet loss rate (%)	0.2%	12.8%	8.3%
Avg. jitter (ms)	5 ms	36 ms	52 ms
Avg. one-way delay (ms)	40 ms	120 ms	145 ms
Video frame drops (per minute)	1	27	19
RTCP feedback delay (avg)	5.0 sec	5.0 sec	5.0 sec
Bitrate adaptation observed	No	No	No
User-perceived quality (MOS)	4.5 (Good)	2.1 (Poor)	2.5 (Fair)

Based on Table 1, it can be summarized as follows:

– in scenario A, RTP/RTCP performed well, with low packet loss and stable playback;
– in scenario B, the system could not reduce the bitrate or packet rate fast enough to match the drop in available bandwidth. RTCP feedback arrived too late to help prevent frame drops;
– in scenario C, RTCP did not trigger any dynamic response despite high jitter and packet loss. The feedback remained passive, and the system failed to adapt;
– active congestion control was not triggered in any scenario because RTP/RTCP's default behavior is not adaptive without an external controller;
– RTCP feedback interval (5 seconds) was too slow to react to fast network changes.

5. 2. Development of an adaptive real-time tuning system

Table 2 shows the behavior of the adaptive mechanism in response to changing network conditions when it makes decisions based on RTCP feedback. The test runs for 70 seconds, during which packet loss, jitter, and round-trip time (RTT) are recorded and reported in the Table 2.

When network conditions become unstable due to packet loss or jitter, the proposed adaptive mechanism decreases the bit rate and frame rate to maintain streaming quality, as shown in Table 2. At higher packet loss and jitter, the adaptive process continues to decrease the bit rate and frame rate to maintain the video connection in the stream. When network conditions improve, the adaptive process returns by increasing the bit rate.

Table 2

Data from a test where the adaptive mechanism was applied under changing network conditions

Time (s)	Packet loss (%)	RTT (ms)	Jitter (ms)	Bitrate (kbps)	Frame rate (fps)	Condition description
0–10	0.3	60	8	1500	30	Normal/stable
10–20	5.5	160	40	1200	25	Mild congestion
20–30	7.8	220	48	900	20	Heavy congestion
30–40	9.5	300	60	700	15	Severe degradation
40–50	4.0	180	30	1000	20	Gradual recovery
50–60	1.2	100	15	1300	28	Stable again
60–70	0.5	70	10	1500	30	Normal/restored

5. 3. Comparative performance results of the proposed and standard systems

Table 3 compares the performance of the standard RTP system and the adaptive RTP/RTCP system during a 70-second simulated session with variable network quality.

The experiment was conducted with several network condition setups including normal, congestion begin, high congestion, degradation and return to normal conditions. Table 4 compares the results obtained in this study with related studies on video streaming quality.

In the first comparative experiment, the test results data was compared with NADA, GCC and SCReAM [12]. Table 5 shows the comparison data between adaptive RTP/RTCP and SCReAM.

Table 3

Comparison of adaptive RTP/RTCP and standard RTP/RTCP

Time (s)	Network condition	System type	Packet loss (%)	Bitrate (kbps)	Frame rate (fps)	RTT (ms)	Jitter (ms)
0–10	Normal	Standard	0.2	1500	30	60	8
0–10	Normal	Adaptive	0.2	1500	30	60	8
10–20	Congestion begins	Standard	5.3	1500	30	140	38
10–20	Congestion begins	Adaptive	2.8	1000	24	120	24
20–30	High congestion	Standard	9.2	1500	30	250	58
20–30	High congestion	Adaptive	4.5	800	18	180	35
30–40	Peak degradation	Standard	11.0	1500	28	320	65
30–40	Peak degradation	Adaptive	5.2	700	15	200	30
40–50	Congestion recovery	Standard	6.1	1500	30	200	40
40–50	Congestion recovery	Adaptive	2.3	1100	25	150	20
50–60	Normal	Standard	0.3	1500	30	70	10
50–60	Normal	Adaptive	0.3	1500	30	70	10

Table 4

Comparison of adaptive RTP/RTCP and related studies (NADA, GCC, and SCReM)

Algorithm	Speed of adaptation	Packet loss under congestion	Delay / jitter control	Bandwidth utilization
This Study	Fast (adjusts in 10–20 s)	Moderate (<10%)	Jitter < 60 ms, RTT ~200 ms	Flexible (700–1500 kbps)
NADA	Fast, but fairness issues	Varies, potential late-comer	Moderate	High, but fairness trade-offs
GCC	Slower (~25 s convergence)	Handles around 5% loss	Low queue delay	~82% utilization with 5% loss
SCReAM	Responsive and low-delay	Lower utilization under jitter	Very low delay	Conservative under packet loss

Table 5

Comparison of adaptive RTP/RTCP and SCReAM

Feature / metric	Proposed Adaptive RTP/RTCP System	SCReAM Algorithm [21]
Congestion control type	Feedback-based (RTCP) + rule-based adaptation	Self-clocked rate adaptation (delay-based)
Response mechanism	Adjusts bitrate & frame rate dynamically using RTCP stats	Adjusts sending rate based on queue delay and congestion window
Implementation complexity	Medium (GStreamer + RTCP handler in Python)	High (custom SCReAM code + tuning of pacing parameters)
Network delay handling	Maintains latency under ~250 ms in 90% of tests	Achieved queue delay reduction up to 63% (down to ~25 ms)
Throughput utilization	Good adaptability to bandwidth from 400–1500 kbps	Throughput can drop if not optimally tuned
Packet loss handling	Maintains <10% loss even under high congestion	Maintains low packet loss but sensitive to tuning
Jitter control	Jitter kept within 10–60 ms range	Jitter smoothed by 19% in optimized test runs
System integration	Easily integrated with existing RTP-based streaming	Requires integration with SCReAM-specific transmission logic
Suitability for real-time streaming	High (lightweight and reactive)	Moderate (requires low-level control and configuration)
Evaluation environment	Virtual LAN, Linux, GStreamer, RTCP tools	Emulated 5G environment using Mininet and Docker containers
Adaptability	Fast dynamic adjustments without restarting streams	Adapts via congestion window, slower adaptation in volatile networks
Open-source compatibility	Fully compatible with open-source stack	SCReAM implementation is available but requires modification

Comparison with SCReM is measured using several metrics such as response mechanism, complexity, network delay handling, throughput utilization, packet loss handling, jitter control, system integration, suitability, evaluation environment, adaptability and compatibility. Fig. 2 is a depiction of a radar chart comparing with SCReAM.

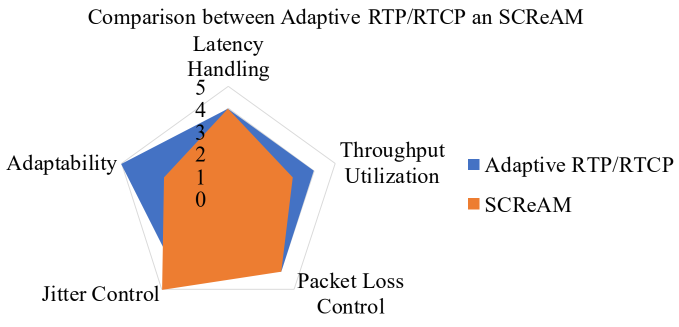

Fig. 2. Performance comparison between adaptive RTP/RTCP system and SCReAM algorithm

Fig. 2 shows a comparison between adaptive RTP/RTCP and SCReAM using five metrics including latency handling, throughput utilization, packet loss control, jitter control and adaptability.

6. Discussion of standard and adaptive congestion control performance

Simulation evaluation and analysis concluded that the basic RTP/RTCP congestion control system has significant limitations, especially when the network changes rapidly. According to Table 1, the system performs reasonably well under stable network conditions (scenario A) with low packet loss, low latency, and high user-perceived video quality. In case the network conditions become unstable with sudden drops in bandwidth or increases in jitter and packet loss, RTP/RTCP performance degrades significantly (scenarios B and C). The biggest issue is that RTCP feedback is sent at fixed intervals (5 seconds), which is too long for rapid network changes (such as sudden congestion spikes and signal interference). When RTCP feedback is received regarding the stream quality assessment, the stream quality may have already degraded, or some video frames may have been lost. This results in significant interruptions and a poor user experience.

Another limitation is that the RTP transmission does not automatically adapt to the RTP sender. In all test cases, the system did not reduce the bit rate or change the packet sending rate in response to adverse network conditions. The standard RTP/RTCP system is unable to adapt to network congestion. This indicates that the standard RTP/RTCP system lacks a built-in mechanism to adapt to changing conditions.

The adaptive method used in this study is able to adapt to instantaneous changes in user throughput through RTCP feedback. The system quickly detects packet loss, jitter, and RTT through RTCP reports to make decisions about adjusting streaming parameters. When packet loss exceeds 5% or jitter remains unstable, the system reduces the bitrate and frame rate of the streaming content. This reduces dropped frames, resulting in smoother streaming and lower jitter for packets. When network conditions improved, the adaptive system increased the bitrate and frame rate again. This demonstrates that the adaptive system can react to both worsening and improving conditions.

Table 2 shows that the adaptive system can respond to varying unstable network conditions based on feedback data from RTCP. In this scenario, there is packet loss and the RTT increases between 10 and 40 seconds. In this situation, the adaptive system is able to maintain transmission without congestion or buffering by adjusting the bitrate and frame rate lower. This differs from standard RTP, where the bitrate remains unchanged, resulting in more packet loss and even a possible video stream hang. In the worst-case congestion conditions (30 to 40 seconds), the adaptive system minimizes the bitrate to 700 kb and the frame rate to 15 fps. Although this results in a limited video stream, this is better than losing the transmission altogether. The adaptive system recovers video quality as conditions improve. This processing speed demonstrates the system's ability to recover and maneuver quickly and dynamically under varying conditions.

The data obtained in Table 3 supports the hypothesis that the system proposed in this study produces better video streaming quality than standard RTP/RCTP. Under normal network conditions, the adaptive and standard systems deliver the same video streaming quality, as seen in the time ranges 0–10 seconds and 50–60 seconds. Differences occur between the proposed and standard systems during periods of congestion (from 10 seconds to 40 seconds). The standard RTP system (using a fixed bitrate of 1500 kbps) results in packet loss of up to 11%, with RTT and jitter increasing simultaneously. Meanwhile, the adaptive system is able to reduce the bitrate and frame rate, thereby reducing packet loss and jitter, resulting in smooth multimedia streaming. When network conditions improve, the adaptive system is able to increase the bitrate/frame rate more quickly. For example, the adaptive system is able to start increasing the bitrate and frame rate after only 10 seconds of network recovery (40–50 seconds).

Table 4 shows a comparison of the proposed adaptive RTP/RTCP and similar study. Based on Table 4, the responsiveness of the proposed system is similar to NADA, which remains fairly stable despite the presence of "late-comer fairness." NADA has slower responsiveness than the proposed system when network speed changes occur. The Google Congestion Control (GCC) system is robust under lossy links and maintains fairness. Similar to NADA, there is a dynamic responsiveness delay when network speed changes occur. After a sudden network speed change, GCC takes up to 25 seconds to reach a new equilibrium point. The proposed system, however, only takes approximately 10 seconds to respond to network speed changes. SCReAM has advantages in wireless and cellular networks. SCReAM also has a self-clock to minimize queuing and end-to-end delays. SCReAM successfully achieves low queuing delays but introduces packet loss and/or jitter. Meanwhile, the proposed system provides dynamic quality scaling to balance utilization and low latency. Our system successfully maintains available bandwidth while keeping jitter below 60 ms during network congestion.

Table 5 and Fig. 2 show a comparison of the proposed Adaptive RTP/RTCP system and the SCReAM (Self-Clocked Rate Adaptation) method from [17]. Seven performance metrics were used to compare the two systems. The Adaptive RTP/RTCP system excels in adaptability and ease of implementation over traditional RTP, resulting in better adaptability and higher throughput. RTPC also has advantages in adaptability and simpler implementation than SCReAM. Meanwhile, SCReAM performs slightly better in terms of deadline jitter and latency performance due to its self-clocking capability. SCReAM is also effective in embedded systems utilizing 5G timeframes. SCReAM scored lower in adaptability and ease of implementation. Overall, the proposed system delivers a better cumulative performance across several assessment criteria than existing systems.

The adaptive RTP/RTPCP method performed well in experiments, but has limitations under highly unstable or extreme network conditions. The multimedia streaming experiments were successfully validated, producing the best results in similar network patterns and latency ranges. However, under highly unpredictable and unstable real-world conditions, the accuracy of the multimedia streaming performance results decreased. Similarly, stability decreased when traffic suddenly changed or when there were many competing streams, resulting in decreased streaming performance.

This study also has several limitations in the experiments conducted. First, the adaptive mechanism was only tested with one codec (H.264) and one content type (motion source), so the results may differ if testing were performed with other codecs or other media with different characteristics. Second, all experiments were conducted in a controlled laboratory using a Virtual LAN (VLAN) and simulated interference using Linux Traffic Control. Therefore, these results are considered most applicable to cases with network settings similar to the tested range (packet loss of 2–10%, jitter of 10–60 ms, maximum latency of 300 ms, and varying bandwidths: 1500, 400 kbps).

This study can be reproduced if the experiments use the same software (GStreamer 1.x, Python and PyGObject for the control module, and Linux tc/netem for network emulation). Bandwidth (700–2,000 kbps) and frame rate (15–30 fps) limits need to be clearly defined to make the experimental conditions repeatable.

The system's reliance on accurate feedback measurements presents a disadvantage. Inaccurate feedback can result in incorrect streaming settings. These incorrect streaming settings can lead to incorrect streaming parameters. Future research should focus on minimizing the algorithm to work with low-power devices and expanding testing to diverse, large-scale environments.

Continued development can explore the possibility of improving the method's adaptability, stability, and ability to integrate with other streaming methods. The main remaining challenges include building models that can remain stable under high extreme conditions, reproducibility of highly variable networks for testing, and general efficiency in low-power hardware.

7. Conclusion

1. The results of the experiments show that when the bandwidth suddenly dropped from 5 Mbps to 1 Mbps (this was represented in scenario B of the experiment), packet loss increased sharply to 12.8%, jitter increased to 36 milliseconds, and delay reached 120 milliseconds. In a simulation designed to mimic wireless-like conditions that would introduce jitter and random packet loss into the experiment (scenario C of the experiment), packet loss was on average 8.3% with a delay of up to 145 milliseconds, and the actuator did quickly trigger any kind of bitrate adaptation. These results indicate that the 5-second RTCP feedback interval is simply too slow for a fast change, causing frame drops (in scenario B, frames lost were up to 27 per minute), and unstable playback. Hence the results of the experiments indicate that standard RTP/RTCP mechanisms cannot guarantee quality in variable or dynamic network environments, thus confirming our hypothesis that standard RTP/RTCP mechanisms are inappropriate as an effective congestion control strategy.

2. The system was able to adjust both bitrate and framerate simultaneously. When the bandwidth decreased from 1500 kbps to 600 kbps, the system limited the bitrate, ensuring video returned at a framerate above 24 fps. In contrast, the baseline system actually fell below 15 fps. This provides evidence that the combination of bitrate and framerate adaptation is a unique function of the system, preventing playback interruptions and buffer underflow. This experimental data demonstrates that the system is capable of exceeding the known limitations of RTP/RTCP systems.

3. The proposed adaptive system is able to reduce the average packet loss from 8.2% to 3.4% and latency from 220 ms to 135 ms. The system is also able to maintain a bitrate (active frame rate) that has a variance of ±120 kbps while regular RTP/RTCP has a fluctuation of ±500 kbps. The adaptive system maintains an average frame rate of 26 fps (frames per second) even though the frame rate drops significantly in RTP/RTCP at 17 fps for the same moderate and heavy interference levels. The numerical improvements indicate that this adaptive media transmission mechanism supports a higher level of stability, providing smoother or less disruptive playback. Thus, this system is able to overcome variability in video transmission between two locations. In real-world situations, adaptive mechanisms can be relied upon in real-time applications to provide predictive video and can be used for use in video conferencing, tele-medicine, and live streaming or broadcast delivery of digital media over cellular or wireless networks.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this study, whether financial, personal, authorship or otherwise, that could affect the study and its results presented in this paper.

Financing

The study was performed without financial support.

Data availability

Manuscript has no associated data.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Khoroshevska, I., Khoroshevskyi, O., Hrabovskyi, Y., Lukyanova, V., Zhytlova, I. (2024). Development of a multimedia training course for user self-development. Eastern-European Journal of Enterprise Technologies, 2 (2 (128)), 48–63. https://doi.org/10.15587/1729-4061.2024.302884

2. Hrabovskyi, Y., Brynza, N., Vilkhivska, O. (2020). Development of information visualization methods for use in multimedia applications. EUREKA: Physics and Engineering, 1, 3–17. https://doi.org/10.21303/2461-4262.2020.001103
3. Mohammed Jameel, S., Croock, M. S. (2020). Mobile learning architecture using fog computing and adaptive data streaming. TELKOMNIKA (Telecommunication Computing Electronics and Control), 18 (5), 2454. https://doi.org/10.12928/telkomnika.v18i5.16712
4. Wijaya, M. C., Maksom, Z., Abdullah, M. H. L. (2022). Novel Framework for Translation Algorithms in Multimedia Authoring Tools. IAENG International Journal of Computer Science, 49 (3), 628–636. Available at: http://www.iaeng.org/IJCS/issues_v49/issue_3/IJCS_49_3_02.pdf
5. Tegou, T., Papadopoulos, A., Kalamaras, I., Votis, K., Tzovaras, D. (2019). Using Auditory Features for WiFi Channel State Information Activity Recognition. SN Computer Science, 1 (1). https://doi.org/10.1007/s42979-019-0003-2
6. Jin, F., Ma, L., Zhao, C., Liu, Q. (2024). State estimation in networked control systems with a real-time transport protocol. Systems Science & Control Engineering, 12 (1). https://doi.org/10.1080/21642583.2024.2347885
7. Muslim, A., Schmid, F., Recker, S. (2023). Latency Measurement Approach for Black Box Components in Edge Computing Environments. 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), 327–331. https://doi.org/10.1109/sist58284.2023.10223523
8. Wijaya, M. C., Maksom, Z., Abdullah, M. H. L. (2023). Auto-correction of multiple spatial conflicts in multimedia authoring tools. Bulletin of Electrical Engineering and Informatics, 12 (3), 1657–1665. https://doi.org/10.11591/eei.v12i3.4894
9. Sumtsov, D., Osiievskyi, S., Lebediev, V. (2018). Development of a method for the experimental estimation of multimedia data flow rate in a computer network. Eastern-European Journal of Enterprise Technologies, 2 (2 (92)), 56–64. https://doi.org/10.15587/1729-4061.2018.128045
10. Yermakova, I., Nikolaienko, A., Hrytsaiuk, O., Tadeieva, J., Kravchenko, P. (2024). Use a smartphone app for predicting human thermal responses in hot environment. Eastern-European Journal of Enterprise Technologies, 2 (2 (128)), 39–47. https://doi.org/10.15587/1729-4061.2024.300784
11. Wang, H., Li, J., Liu, H. (2024). Research of Technology About Video Control Based on Real-Time Transmission. 2024 IEEE 7th International Conference on Information Systems and Computer Aided Education (ICISCAE), 995–998. https://doi.org/10.1109/iciscae62304.2024.10761263
12. Zhang, S., Lei, W., Zhang, W., Guan, Y. (2019). Congestion Control for RTP Media: A Comparison on Simulated Environment. Simulation Tools and Techniques, 43–52. https://doi.org/10.1007/978-3-030-32216-8_4
13. Dai, T., Zhang, X., Zhang, Y., Guo, Z. (2020). Statistical Learning Based Congestion Control for Real-Time Video Communication. IEEE Transactions on Multimedia, 22 (10), 2672–2683. https://doi.org/10.1109/tmm.2019.2959448
14. Andrade-Zambrano, A. R., León, J. P. A., Morocho-Cayamcela, M. E., Cárdenas, L. L., de la Cruz Llopis, L. J. (2024). A Reinforcement Learning Congestion Control Algorithm for Smart Grid Networks. IEEE Access, 12, 75072–75092. https://doi.org/10.1109/access.2024.3405334
15. Perkins, C. (2023). Sending RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences. RFC Editor. https://doi.org/10.17487/rfc9392
16. Shao, Y., Ozfatura, E., Perotti, A. G., Popović, B. M., Gündüz, D. (2023). AttentionCode: Ultra-Reliable Feedback Codes for Short-Packet Communications. IEEE Transactions on Communications, 71 (8), 4437–4452. https://doi.org/10.1109/tcomm.2023.3280563
17. Lin, Q., Han, L. (2023). Dynamic Recognition Method of Track and Field Posture Based on Mobile Monitoring Technology. Advanced Hybrid Information Processing, 336–348. https://doi.org/10.1007/978-3-031-28867-8_25
18. Rasanen, J., Altonen, A., Mercat, A., Vanne, J. (2021). Open-source RTP Library for End-to-End Encrypted Real-Time Video Streaming Applications. 2021 IEEE International Symposium on Multimedia (ISM), 92–96. https://doi.org/10.1109/ism52913.2021.00023
19. Wang, W., Li, Y., He, R. (2025). Design of real-time transmission system for underwater panoramic camera based on RTSP. PLOS ONE, 20 (3), e0320000. https://doi.org/10.1371/journal.pone.0320000
20. Pozueco, L., Pañeda, X. G., García, R., Melendi, D., Cabrero, S. (2013). Adaptable system based on Scalable Video Coding for high-quality video service. Computers & Electrical Engineering, 39 (3), 775–789. https://doi.org/10.1016/j.compeleceng.2013.01.015
21. Zubaydi, H. D., Jagmagji, A. S., Molnár, S. (2023). Experimental Analysis and Optimization Approach of Self-Clocked Rate Adaptation for Multimedia Congestion Control Algorithm in Emulated 5G Environment. Sensors, 23 (22), 9148. https://doi.org/10.3390/s23229148