

DAMPAK *EDGE CACHING* TERHADAP RELIABILITAS PROTOKOL PENGIRIMAN AUDIO UNTUK ASSESSMENT ONLINE BERSKALA KECIL

Iwan Haryatno^{1*}, M. Syaiful Amin²

^{1,2}Universitas Amikom Purwokerto; Jl. Letjend Pol. Soemarto No.126, Watumas, Purwanegara, Kec. Purwokerto Utara, Kabupaten Banyumas, Jawa Tengah 53127

Keywords:

content delivery network;
HTTP 206 Partial Content;
HTTP Live Streaming;
audio assessment;
jaringan mobile Indonesia

Correspondent Email:

iwanharyatno1@gmail.com

Abstrak. Penelitian ini menginvestigasi dampak *edge caching* terhadap reliabilitas pengiriman audio *assessment online* berskala kecil melalui perbandingan empat konfigurasi: Direct No-CDN, Direct CDN, HLS No-CDN, dan HLS CDN. Konteks ini mengevaluasi audio *prerecorded* berdurasi pendek (3–13 detik) pada jaringan mobile Indonesia dengan 235 pengguna riil menggunakan desain *within-subjects* dan analisis sensitivitas *threshold Acceptable Delay* (0 ms, 500 ms, 3000 ms). Hasil Cochran's Q menunjukkan efek signifikan konfigurasi terhadap reliabilitas ($Q = 265,88$, $p < 0,001$). CDN mengurangi tingkat gagal AD dari ~75–82% menjadi ~14–21% secara independen dari protokol. Uji *Mann-Whitney U* mengkonfirmasi tidak terdapat perbedaan signifikan pada median gagal AD antara Direct CDN dan HLS CDN pada threshold 500 ms ($p = 0,706$, median keduanya 0%). Temuan ini menyanggah stereotip bahwa HLS kurang cocok untuk konten berdurasi pendek dan mengidentifikasi *edge caching* sebagai determinan utama reliabilitas, menggeser prioritas dari pilihan protokol ke implementasi CDN.



Copyright © 2026 The Author(s),
Published by Jurnal Informatika dan
Teknik Elektro Terapan. This is an
open-access article distributed under
the terms of the Creative Commons
Attribution-NonCommercial 4.0
International License (CC BY-NC
4.0).

Abstract. This study investigates the impact of edge caching on the reliability of small-scale online audio assessment delivery by comparing four configurations: Direct No-CDN, Direct CDN, HLS No-CDN, and HLS CDN. The context evaluates short-duration prerecorded audio (3–13 seconds) on Indonesian mobile networks with 235 real users using a within-subjects design and sensitivity analysis of Acceptable Delay threshold (0 ms, 500 ms, 3000 ms). Cochran's Q results show a significant configuration effect on reliability ($Q = 265.88$, $p < 0.001$). CDN reduces AD failure rates from ~75–82% to ~14–21% independently of protocol. Mann-Whitney U confirms no significant difference in median AD failure rate between Direct CDN and HLS CDN at 500 ms threshold ($p = 0.706$, both medians 0%). These findings challenge the stereotype that HLS is inherently unsuitable for short-duration content and identify edge caching as the primary reliability determinant, shifting priority from protocol choice to CDN implementation.

1. PENDAHULUAN

Penilaian bahasa Inggris berbasis komputer telah menjadi paradigma dominan dalam pendidikan modern, namun pertumbuhan platform *assessment online* memperkenalkan tantangan teknis signifikan pada lapisan jaringan. Uddin et al. [1] menunjukkan bahwa latensi, *rebuffering*, dan stabilitas *bitrate* sangat

mempengaruhi QoE pada jaringan variabel, yang dalam konteks *assessment* dapat mengkompromikan validitas tes. Taraghi et al. [2] mengkonfirmasi bahwa *stalling* merupakan impairment QoE paling signifikan, bahkan lebih merusak daripada variasi kualitas *bitrate*. Untuk konten audio berdurasi 3–13 detik dalam *assessment listening*, setiap keterlambatan

pengiriman yang melebihi durasi konten dapat mengakibatkan kegagalan fatal bagi pengalaman peserta tes. Lebih lanjut penelitian El-saleh et al. [3] pada konteks jaringan seluler Indonesia yang dikarakterisasi oleh variabilitas latensi, *jitter*, dan *packet loss* yang signifikan membuat kondisi ini menjadi semakin kritis.

Dalam literatur teknis, terdapat konsensus mapan bahwa HTTP *Adaptive Streaming* (HLS/DASH) menjadi standar dominan untuk konten berdurasi panjang dengan kemampuan adaptasi *bitrate* dinamis. Timmerer et al. [4] menyatakan HLS dirancang untuk segmentasi dengan durasi segmen 6–10 detik (standar) atau 1–2 detik (*low-latency*), memungkinkan klien beralih antar-representasi *bitrate* secara dinamis. Uddin et al. [1] mengevaluasi algoritma *low-latency* ABR dan menemukan optimasi *bitrate* hingga 2149 kbps dengan variabilitas minimal pada segmen 2–10 detik. Namun, Zhang et al. [5] mengidentifikasi bahwa *low-latency* HLS dengan segmen 4 detik dan fragment 1 detik memperkenalkan latensi struktural dari *manifest parsing* dan *segmented request*, yang menjadi besar secara proporsional ketika durasi konten mendekati atau lebih pendek dari durasi segmen. Literatur tersebut eksklusif berfokus pada konten berdurasi menengah hingga panjang, tanpa mempertimbangkan fenomena *sub-segment-duration mismatch* — kondisi di mana durasi konten (3–13 detik) lebih pendek dari durasi segmen standar (4 detik), menyebabkan *overhead* protokol menjadi lebih besar secara proporsional.

Berdasarkan pemahaman di atas, praktik industri membentuk stereotip kuat: untuk konten audio statis berdurasi pendek, *progressive download* berbasis HTTP 206 *Partial Content* dianggap "cukup baik" dan lebih sederhana. Nie et al. [6] menunjukkan *zero-copy* mengurangi volume transmisi data 6,25 kali dan downtime 31,8%, namun berfokus pada migrasi kontainer cloud. Zhou dan Lin [7] mengkonfirmasi *memory copying* antar *kernel-user boundary* mengonsumsi hingga 55,6% total siklus CPU, menjadikan *zero-copy* kritis untuk sistem *high throughput*.

Stereotip tersebut mengandung kebenaran, namun tidak lengkap. Liotou et al. [8] mengevaluasi DASH dengan *caching* dan menunjukkan *cache hit* mengurangi *stall* secara signifikan, namun terbatas pada video berdurasi

panjang dengan segmen 5 detik. Dalam konteks sistem *assessment* berskala kecil, CDN seringkali diabaikan atau dianggap sebagai *over-engineering* karena jumlah pengguna yang relatif rendah. Pengaruh CDN terhadap reliabilitas pengiriman *audio assessment* berdurasi 3–13 detik—yang secara struktural lebih pendek dari durasi segmen standar HLS—belum terdokumentasi dalam literatur *peer-reviewed*.

Tantangan yang teridentifikasi adalah bahwa literatur teknis didominasi oleh solusi *enterprise-scale* untuk *video live streaming* dengan jutaan pengguna simultan, sementara konteks *assessment* online berskala kecil dengan konten *prerecorded* statis berdurasi pendek tidak mendapat perhatian proporsional. Waheed et al. [9] mengusulkan arsitektur *adaptive bitrate streaming* dengan CDN yang mengintegrasikan *load balancer* dan *caching server*, menunjukkan bahwa penggunaan CDN mengurangi *response time* kumulatif dari 3.200.000 ms menjadi 200.000 ms pada skenario 150 pengguna simultan. Namun, evaluasi tersebut berfokus pada video berdurasi panjang dengan segmen 2–20 detik, tanpa mempertimbangkan konten *sub-segment-duration*. Tidak terdapat studi yang secara sistematis membandingkan protokol HTTP 206 dan HLS dalam konfigurasi yang identik — dengan dan tanpa CDN — pada jaringan mobile di negara berkembang.

Berdasarkan kesenjangan tersebut, penelitian ini menginvestigasi: Sejauh mana *edge caching* mempengaruhi reliabilitas pengiriman audio *assessment* dibandingkan pilihan protokol terutama HTTP 206 *Partial Content* vs. HLS pada jaringan mobile Indonesia? Penelitian ini tidak menguji kemampuan penuh *adaptive bitrate streaming*, melainkan mengisolasi dampak infrastruktur CDN sebagai moderator dalam perbandingan protokol untuk konten audio statis berdurasi *sub-segment* (3–13 detik). Pertanyaan diformulasikan: (1) Bagaimana karakteristik reliabilitas pada masing-masing empat konfigurasi? (2) Sejauh mana CDN mengurangi tingkat kegagalan pada masing-masing protokol? (3) Apakah Direct dengan CDN menyediakan reliabilitas sebanding dengan HLS dengan CDN?

Kontribusi utama penelitian ini adalah: (1) Karakterisasi empiris dampak CDN terhadap

reliabilitas pengiriman *audio assessment*; (2) Demonstrasi equivalensi protokol dalam kondisi CDN (3) Validasi *threshold* toleransi *delay* praktikal; dan (4) Landasan empiris bagi pengembang sistem *assessment* skala kecil terkait pertimbangan investasi *edge caching* dan kompleksitas protokol *adaptive streaming*.

2. TINJAUAN PUSTAKA

2.1. Virtual Private Server (VPS)

Virtual Private Server (VPS) adalah bentuk *multi-tenant cloud hosting* dimana sumber daya server virtual disediakan bagi pengguna melalui internet. Setiap VPS beroperasi pada mesin fisik yang menjalankan beberapa unit virtual, di mana masing-masing unit memiliki sistem operasi dan sumber daya terdedikasi, mencakup memori, komputasi, dan *bandwidth* CPU [10]. VPS menawarkan keseimbangan antara performa, fleksibilitas, dan kontrol di antara *shared hosting* dan *dedicated hosting*, menjadikannya solusi *cost-effective* untuk aplikasi dengan trafik moderat. Penelitian [11] mengonfirmasi bahwa dalam lingkungan dengan keterbatasan sumber daya, Nginx dalam VMware CentOS 7 dengan *1-core 1-thread* CPU, 2GB *memory* dan 40G *harddisk* menghasilkan performa superior, menunjukkan kesesuaiannya untuk *deployment* skala terbatas.

2.2. Nginx Web Server

Nginx dirancang dengan arsitektur *event-driven* yang memungkinkannya menangani ratusan ribu koneksi konkuren pada perangkat keras modern. Berbeda dengan server web tradisional berbasis *threaded* atau *process-based*, Nginx menggunakan satu *worker process* per *core* untuk menangani beberapa koneksi secara efisien [12]. Penelitian [13] menunjukkan bahwa model Nginx mencapai *throughput* dan *connection time* yang jauh lebih baik dibandingkan Apache. Terkait protokol *streaming*, [14] mencatat bahwa operasi *copy* sederhana pada Nginx hanya mengonsumsi 0.7-1.0% CPU, dengan latensi terukur sebesar 8 detik untuk HLS dan 4 detik untuk DASH.

2.3. Node.js Express API

Node.js Express API merupakan teknologi *backend* berarsitektur *event-driven* dan *non-blocking I/O* yang sangat efisien untuk menangani operasi data *I/O-bound* seperti

fungsi CRUD. Riset komparatif membuktikan Node.js memiliki performa superior dibanding PHP Laravel dengan capaian *throughput* hingga 422,6 RPS atau 2,2 kali lipat lebih tinggi [15]. Selain unggul dalam kecepatan pemrosesan, arsitektur ini sangat hemat sumber daya dengan konsumsi CPU server yang stabil di kisaran 11% sehingga sangat andal untuk mengelola data telemetri secara *real-time*.

2.4. HTTP Chunked Delivery dan Partial Content

HTTP chunked transfer encoding (RFC 9112 §7.1) adalah mekanisme pengiriman data dalam serangkaian *chunk* yang masing-masing diawali oleh ukuran heksadesimalnya. Hal ini memungkinkan server memulai pengiriman tanpa harus mengetahui ukuran akhir respons [16]. Di sisi lain, *HTTP 206 Partial Content* merupakan kode status respons yang mengindikasikan bahwa server berhasil mengirimkan rentang data (*range*) yang diminta melalui header "Range" [17].

2.5. Content Delivery Network (CDN)

Content Delivery Network (CDN) adalah infrastruktur terdistribusi secara geografis yang terdiri dari *edge server* untuk mengirimkan konten digital secara efektif [18]. CDN mengoptimalkan pengiriman melalui *caching* pada *edge servers* yang lebih dekat dengan pengguna, sehingga mengurangi latensi dan beban pada *origin server*. Meskipun [19] menganalisis latensi rendah pada *edge computing* menggunakan protokol AODV, riset tersebut lebih berfokus pada optimasi rute perangkat IoT daripada evaluasi *cost-benefit* pada skala ratusan pengguna simultan.

2.6. W3C Resource Timing API

Resource Timing API merupakan spesifikasi standar yang memungkinkan aplikasi web mengakses informasi timing lengkap untuk setiap resource dalam dokumen, termasuk fase-fase granular dari siklus hidup permintaan jaringan [20]. Sistem ini bertugas menangkap siklus hidup setiap permintaan jaringan secara granular sejak permintaan diinisialisasi hingga data diterima sepenuhnya oleh peramban pengguna. Siklus jaringan tersebut terdiri atas durasi *Stalled*, DNS, TCP, dan SSL *handshake* untuk memantau beban koneksi, serta *Waiting Time* (TTFB) dan

Content Download untuk mengukur kecepatan transmisi payload data.

2.7. Toleransi Delay pada Audio Streaming

Definisi *acceptable delay* pada penelitian ini menggunakan *threshold* 500 ms untuk membedakan antara keterlambatan pengiriman yang masih dapat ditoleransi dan kegagalan sistem yang signifikan dalam konteks *assessment audio* non-interaktif. Dikmen et al. [21] mengembangkan skala kecemasan tes online dan mengidentifikasi faktor *technical anxiety* ($\alpha = 0,89$) yang mencakup ketakutan akan gangguan koneksi internet selama *assessment*, mengkonfirmasi bahwa ketidakpastian teknologi berkontribusi pada kecemasan peserta tes. Di sisi teknis, ITU-T G.114 [22] merekomendasikan bahwa *one-way delay* hingga 150 ms masih transparan untuk aplikasi interaktif, sementara delay 150–400 ms mulai mempengaruhi kualitas percakapan namun tetap dapat diterima. Tsioutas et al. (2025) menunjukkan bahwa pada *Network Music Performance* (NMP) musisi masih dapat mensinkronkan performa dengan *one-way delay* hingga 40 ms, melebihi *threshold* 25–30 ms yang sebelumnya dianggap batas toleransi.

Mengingat *assessment listening* merupakan aktivitas non-interaktif tanpa referensi visual dan tanpa kebutuhan sinkronisasi real-time, toleransi delay yang lebih tinggi merupakan kompromi yang konservatif. Batas bawah *strict* (0 ms) dipilih untuk merepresentasikan kondisi ideal tanpa toleransi, mengisolasi overhead struktural protokol murni tanpa kompensasi jaringan. Batas atas *lenient* (3000 ms) dipilih berdasarkan temuan Li et al. [23] yang menunjukkan bahwa pengguna mulai meninggalkan video ketika *startup delay* melebihi 2 detik, dengan setiap penambahan 1 detik meningkatkan *abandonment rate* sebesar 5,8%.

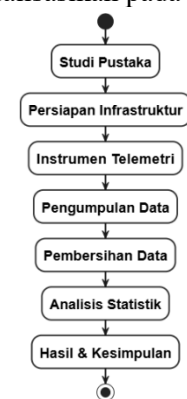
2.8. Desain Eksperimental *Within-Subject*

Desain eksperimental *within-subject* adalah jenis metodologi penelitian di mana setiap peserta dihadapkan pada semua tingkat variabel independen [24]. Pendekatan ini memungkinkan peneliti untuk mengontrol variabel independen pada level individu karena setiap subjek berfungsi sebagai kontrol bagi dirinya sendiri, sejalan dengan karakteristik fundamental desain *within-subjects* yang

menghilangkan efek variasi antar individu. Kombinasi pendekatan ini dengan mekanisme *counterbalancing* yang bertujuan untuk meniadakan *sequence bias* atau bias urutan, di mana performa sistem mungkin terlihat menurun di akhir sesi bukan karena faktor jaringan, melainkan akibat faktor eksternal seperti *thermal throttling* pada perangkat pengguna atau beban panas pada server. Dalam konteks *causal inference*, *sequential randomization* pada desain *counterbalanced* terbukti mampu menghasilkan *unbiased estimates* terlepas dari spesifikasi model [25].

3. METODE PENELITIAN

Penelitian ini menggunakan Metode Eksperimen Kuantitatif dengan melakukan pengujian langsung terhadap performa distribusi data *Direct Streaming* (HTTP 206) dan *HTTP Live Streaming* (HLS) pada lingkungan *web server* Nginx. Pendekatan ini bertujuan untuk menguji hubungan sebab-akibat secara empiris dengan memanipulasi variabel jenis protokol dan arsitektur *edge caching* pada pengujian pengguna riil (*Real User Monitoring*). Tahapan penelitian secara sistematis divisualisasikan pada Gambar 3.1.



Gambar 3.1. Diagram Alir Penelitian

Gambar 3.1 menunjukkan tahapan pelaksanaan penelitian yang dibagi menjadi tujuh langkah keseluruhan sebagai berikut:

3.1. Studi Pustaka

Melakukan eksplorasi teoritis dari berbagai literatur ilmiah terdahulu mengenai arsitektur pengiriman objek multimedia untuk menetapkan parameter teknis acuan. Tahap ini berfokus pada penentuan batas *bitrate* standar (128 kbps CBR), durasi pemotongan segmen audio HLS sepanjang 4 detik yang merupakan

default dari *Apple HLS Tools* [5] serta perumusan metrik *Acceptable Delay* (AD) sebagai variabel dependen utama untuk evaluasi reliabilitas praktis.

3.2. Persiapan Infrastruktur

Server *backend* menggunakan VPS yang memiliki spesifikasi 4 vCPU, 8GB RAM dan 150 GB SSD. Infrastruktur backend untuk pengumpulan data telemetri itu sendiri menggunakan Node.js dengan *framework* Express.js yang sangat efisien untuk operasi I/O *bound* seperti CRUD. Selain itu, integrasi *edge caching* dilakukan melalui skema Cloudflare Proxy guna membagi lingkungan eksperimen ke dalam dua kondisi infrastruktur yang identik, yaitu jaringan dengan CDN dan jaringan tanpa CDN (*Non-CDN*).

Implementasi HTTP 206 *Partial Content* menggunakan Server Nginx yang dikonfigurasi dengan *sendfile on*, *tcp_nopush on*, dan *tcp_nodelay on* untuk mengoptimalkan zero-copy transfer. File audio MP3 dengan *bitrate* 128 kbps CBR disimpan sebagai *single file* dan disajikan melalui HTTP 206 *Partial Content* sesuai RFC 7233. Klien mengirimkan *header* Range untuk mengambil bagian file yang diperlukan, dan server merespons dengan kode status 206 beserta *byte range* yang diminta.

Implementasi HLS mencakup konversi file audio yang identik (MP3, 128 kbps CBR) ke format HLS menggunakan Ffmpeg, konfigurasi konversi yang digunakan menghasilkan: Playlist *master* (.m3u8) dengan daftar segmen, Segmen audio (.ts) dengan durasi 4 detik, *encoding* identik dengan *file Direct*, tanpa *multi-bitrate ladder* (single quality), dan tanpa *adaptive bitrate algorithm* pada sisi klien. Pemilihan durasi segmen 4 detik merefleksikan konfigurasi *default* pada *Apple HLS Tools* [5], bukan optimalisasi khusus untuk konten berdurasi pendek. Konfigurasi ini sengaja dibatasi pada *single-bitrate* untuk mengisolasi *overhead* struktural dari arsitektur segmentasi dan *manifest-based delivery* dari manfaat *adaptive bitrate switching*.

3.3. Pengembangan Instrumen Telemetri

Membangun modul pelacak performa otomatis di sisi klien (*client-side*) menggunakan *custom hook* berbasis kerangka kerja React.js. Modul telemetri ini memanfaatkan spesifikasi terstandarisasi W3C *Resource Timing API*

untuk menangkap metrik waktu jaringan secara granular sejak permintaan diinisialisasi, yang mencakup fase *stalled*, jabat tangan (*handshake* DNS/TCP/SSL), *Waiting Time* (TTFB), hingga durasi unduh *payload*.

Metrik *acceptable delay* (AD) secara matematis didefinisikan sebagai:

$$AD = total\ time - chunk\ duration > 500$$

3.4. Pengumpulan Data

Melaksanakan akuisisi data performa di lapangan melalui skenario ujian mendengarkan (*listening assessment*) aktual secara serentak pada platform *elobright.com*. Tahap pengumpulan data ini mengandalkan interaksi langsung dari 263 data mentah partisipan aktif yang mengakses konten secara *within-subjects* untuk mendapatkan log telemetri jaringan orisinal yang merepresentasikan fluktuasi operator seluler di Indonesia.

3.5. Pembersihan Data

Melakukan prapemrosesan terhadap dataset mentah yang diekspor dari basis data server guna memitigasi bias dan variabel perancu (*confounding variable*). Langkah pembersihan data ini dijalankan melalui tiga prosedur utama, yaitu penghapusan entri log ganda (*deduplication*), eksklusi file audio instruksi/konteks berdurasi panjang, serta penerapan metode *listwise deletion* untuk menjamin keseimbangan sampel data. Proses deduplikasi menghasilkan 235 partisipan unik yang juga mengalami semua konfigurasi, sementara file audio akhir yang digunakan memiliki rentang durasi antara 3-13 detik sejumlah 10 item dengan 5 file audio format MP3 dengan *bitrate* 128 kbps CBR dan 5 audio HLS *encoding*.

3.6. Analisis Statistik Inferensial

Menguji hipotesis dan tingkat signifikansi perbedaan performa dari manipulasi arsitektur data secara kuantitatif menggunakan perangkat lunak statistik. Validasi temuan dilakukan secara global melalui uji Cochran's Q untuk metrik biner *Acceptable Delay* (AD) dengan $df = 3$. Analisis *post-hoc* menggunakan uji McNemar dengan koreksi Bonferroni untuk Cochran's Q. Selain itu, validasi konsistensi per-pengguna dilakukan melalui uji *chi-square* terhadap tabel kontingensi kategori *Acceptable*

Delay (Tidak Pernah/Kadang-Kadang/Selalu AD) untuk menguji apakah distribusi pola kegagalan secara signifikan bergantung pada konfigurasi arsitektur. Uji ini mengonfirmasi bahwa perbedaan reliabilitas antar protokol bersifat sistematis dan terkait karakteristik protokol, bukan variabilitas jaringan yang acak.

3.7. Penarikan Hasil dan Kesimpulan

Menyusun interpretasi logis dari hasil pengolahan data statistik untuk menjawab rumusan masalah serta tujuan penelitian mengenai efisiensi arsitektur data audio. Hasil pengujian divisualisasikan ke dalam bentuk grafik *network waterfall*, *box plot*, dan tabel komparasi berpasangan, yang kemudian diringkas menjadi poin kesimpulan operasional serta lembar saran bagi pengembang sistem informasi di masa depan.

4. HASIL DAN PEMBAHASAN

4.1. Uji Signifikansi Global

Berdasarkan uji Cochran's Q terhadap metrik *Acceptable Delay* (AD) dengan *threshold* toleransi 500 ms dari 235 pengguna lengkap ($df = 3$), diperoleh hasil yang sangat signifikan ($Q = 265,88$, $p < 0,001$), mengindikasikan bahwa konfigurasi pengiriman audio secara signifikan mempengaruhi reliabilitas praktis pemutaran. Efek ukuran praktis ditunjukkan oleh reduksi gagal AD *rate* dari 81,7% (Direct No-CDN) menjadi 14,0% (Direct CDN), sebanyak 67,7 poin persentase setara dengan reduksi relatif 82,9%.

Uji *post-hoc* McNemar dengan koreksi Bonferroni mengonfirmasi bahwa seluruh pasangan konfigurasi berbeda secara signifikan ($p_{\text{bonf}} < 0,05$), dengan perbedaan terbesar terdapat pada dimensi CDN (Direct No-CDN vs Direct CDN: $p_{\text{bonf}} = 5,18 \times 10^{-34}$, 138 vs 5 *discordant*) dibandingkan dimensi protokol (Direct CDN vs HLS CDN: $p_{\text{bonf}} = 1,39 \times 10^{-2}$, 6 vs 23 *discordant*). Temuan ini mengindikasikan bahwa pengaruh CDN terhadap reduksi kegagalan sistem jauh lebih dominan dibandingkan pilihan protokol, dan perbedaan protokol hanya menjadi signifikan ketika CDN tidak diaktifkan.

4.2. Pola *Acceptable Delay* dengan *threshold* 500ms

Dengan desain *within-subjects*, setiap peserta mengalami keempat konfigurasi pengiriman audio. Analisis konsistensi per-pengguna mengungkapkan pola kegagalan yang fundamental berbeda antar protokol, sebagaimana disajikan pada Tabel 4.1 dan 4.2.

Tabel 4.1. Pola Konsistensi *Acceptable Delay* Per-Pengguna. TGAD = Tidak pernah gagal AD; KGAD=Kadang-kadang gagal AD; SGAD=Selalu gagal AD

Konfigurasi	TGAD (%)	KGAD (%)	SGAD (%)
Direct No-CDN	5,5	25,6	68,9
Direct CDN	68,4	28,1	3,6
HLS No-CDN	2,8	55,1	42,1
HLS CDN	73,2	11,4	15,4

Tabel 4.2. Statistik Deskriptif Tingkat Kegagalan AD Per-Pengguna.

Konfigurasi	Rata-Rata Gagal AD (%)	Median Gagal AD (%)
Direct No-CDN	81,7	100,0
Direct CDN	14,0	0,0
HLS No-CDN	74,9	66,7
HLS CDN	21,1	0,0

Temuan ini secara langsung menyanggah stereotip yang diuraikan pada Bagian Pendahuluan bahwa *progressive download* berbasis HTTP 206 secara otomatis menjadi solusi yang "cukup baik" untuk konten audio berdurasi pendek. Data menunjukkan bahwa tanpa CDN, Direct No-CDN memiliki 68,9% pengguna yang SGAD (median 100%) — angka yang bahkan lebih buruk dari HLS No-CDN (42,1%). Stereotip yang menyatakan *Direct* lebih unggul untuk konten pendek hanya berlaku ketika infrastruktur *edge caching* tersedia; tanpanya, kedua protokol mengalami kegagalan sistematis yang ekstrem. Hal ini mengindikasikan bahwa untuk jaringan *mobile* Indonesia dengan karakteristik variabilitas tinggi, *edge caching* merupakan komponen kritis bagi reliabilitas sistem, terlepas dari pilihan protokol.

Ketika CDN diaktifkan, pola kegagalan berubah secara fundamental. Direct CDN

mengalami penurunan drastis pada kategori SGAD dari 68,9% menjadi 3,6%, dengan median gagal AD *rate* turun dari 100% menjadi 0%. HLS CDN menunjukkan perbaikan yang lebih moderat namun tetap substantif: kategori SGAD turun dari 42,1% menjadi 15,4%, dengan median yang juga turun menjadi 0%. Lebih menarik lagi, HLS CDN menunjukkan proporsi TGAD yang lebih tinggi (73,2%) dibandingkan Direct CDN (68,4%), mengindikasikan bahwa protokol segment-based dengan CDN dapat memberikan reliabilitas yang sebanding — bahkan sedikit lebih konsisten — untuk konten berdurasi pendek, kontra-intuitif terhadap literatur yang menyatakan HLS kurang cocok untuk skenario tersebut.

Uji Chi-square terhadap tabel kontingensi kategori konsistensi menunjukkan bahwa distribusi gagal AD secara signifikan bergantung pada konfigurasi ($\chi^2 = 576,31$, $p = 3,00 \times 10^{-121}$, $df = 6$). Uji *Mann-Whitney U* antara Direct CDN dan HLS CDN mengonfirmasi tidak terdapat perbedaan signifikan pada median gagal AD *rate* ($U = 30631,00$, $p = 0,706$), dengan median kedua konfigurasi sebesar 0%. Temuan ini mengkonfirmasi bahwa pada *threshold* toleransi 500 ms, kedua protokol dengan CDN menyediakan reliabilitas yang sebanding pada tingkat median, mengesampingkan klaim bahwa HLS secara inheren kurang cocok untuk konten berdurasi pendek.

Stereotip "Direct untuk pendek, HLS untuk panjang" adalah oversimplifikasi yang tidak memperhitungkan konteks infrastruktur jaringan. Tanpa CDN, kedua protokol gagal; dengan CDN, kedua protokol berhasil. Perbedaan protokol hanya menjadi signifikan pada kondisi tanpa CDN, di mana *overhead* struktural HLS memang lebih parah. Namun, ketika CDN diaktifkan, perbedaan tersebut menjadi *negligible*, menggeser faktor kritis dari pilihan protokol ke investasi *edge caching*.

4.3. Pola Acceptable Delay dengan Threshold Ideal 0 ms

Analisis sensitivitas dengan *threshold strict* (0 ms) merepresentasikan kondisi ideal tanpa toleransi, mengisolasi *overhead* struktural protokol murni tanpa kompensasi jaringan. Pada *threshold* ini, definisi gagal AD menjadi *total_time_timing > chunk_duration_ms*, di

mana setiap keterlambatan pengiriman yang melebihi durasi konten langsung dikategorikan sebagai kegagalan sistem.

Tabel 4.3. Pola Konsistensi *Acceptable Delay* Per-Pengguna. TGAD = Tidak pernah gagal AD; KGAD=Kadang-kadang gagal AD; SGAD=Selalu gagal AD

Konfigurasi	TGAD (%)	KGAD (%)	SGAD (%)
Direct No-CDN	1,7	25,1	73,2
Direct CDN	1,7	69,4	28,9
HLS No-CDN	0,0	0,9	99,1
HLS CDN	0,4	1,3	98,3

Tabel 4.4. Statistik Deskriptif Tingkat Kegagalan AD Per-Pengguna.

Konfigurasi	Rata-Rata Gagal AD (%)	Median Gagal AD (%)
Direct No-CDN	85,7	100,0
Direct CDN	57,7	33,3
HLS No-CDN	99,7	100,0
HLS CDN	98,9	100,0

Temuan pada *threshold* 0 ms mengungkapkan pola kegagalan yang ekstrem dan terkonsentrasi pada protokol HLS. HLS No-CDN menunjukkan 99,1% pengguna yang selalu gagal AD (median 100%), sementara HLS CDN hampir tidak menunjukkan perbaikan dengan 98,3% pengguna yang SGAD (median 100%). Fenomena ini secara langsung mengkonfirmasi adanya *overhead struktural* pada protokol *segment-based* — *manifest parsing* dan *segmented request* memperkenalkan latensi yang secara konsisten melebihi durasi konten 3–13 detik, terlepas dari aktivasi CDN. Sejalan dengan temuan Zhang et al. [5], segmentasi 4 detik menciptakan *sub-segment-duration mismatch* di mana *overhead* protokol menjadi proporsional lebih besar daripada durasi konten itu sendiri.

Sebaliknya, Direct CDN menunjukkan distribusi yang fundamental berbeda: kategori SGAD turun dari 73,2% menjadi 28,9%, dengan mayoritas pengguna (69,4%) berada pada kategori KGAD dan median gagal AD *rate* turun menjadi 33,3%. Distribusi *right-skewed*

ini (mean 57,7% > median 33,3%) mengindikasikan bahwa kegagalan pada Direct CDN bersifat sporadis dan terkait variabilitas jaringan, bukan karakteristik sistemik dari protokol. CDN berhasil mengkompensasi sebagian besar *overhead* jaringan untuk protokol *progressive download*, namun tidak dapat mengkompensasi *overhead* struktural HLS.

Uji *Chi-square* menunjukkan bahwa distribusi gagal AD secara signifikan bergantung pada konfigurasi ($\chi^2 = 410,42$, $p = 1,61 \times 10^{-85}$, $df = 6$). Uji *Mann-Whitney U* antara Direct CDN dan HLS CDN mengkonfirmasi perbedaan yang sangat signifikan ($U = 8409,50$, $p = 5,78 \times 10^{-53}$), dengan median gagal AD rate Direct CDN sebesar 33,3% dibandingkan 100% untuk HLS CDN. Perbedaan median yang ekstrem ini mengkonfirmasi bahwa pada *threshold* 0 ms, HLS mengalami kegagalan sistematis yang universal, sementara Direct CDN menyediakan reliabilitas yang signifikan bagi sebagian pengguna.

Perbandingan antara *threshold* 0 ms dan 500 ms mengonfirmasi bahwa: pada 0 ms, HLS gagal secara universal (98–99%) sementara Direct CDN menunjukkan variabilitas (median 33,3%); pada 500 ms, perbedaan tersebut menyusut hingga tidak signifikan secara statistik (median keduanya 0%). Fenomena ini mengkonfirmasi bahwa *overhead* struktural HLS memang ada dan dapat diukur, namun menjadi *negligible* pada toleransi praktis 500 ms ketika CDN diaktifkan. Temuan ini secara langsung menyanggah stereotip bahwa HLS secara inheren tidak cocok untuk konten berdurasi pendek — pada kondisi CDN, kedua protokol menyediakan reliabilitas yang sebanding; pada kondisi tanpa CDN, kedua protokol gagal, dengan HLS gagal secara lebih sistematis.

4.4. Pola *Acceptable Delay* dengan *threshold* 3000ms

Analisis sensitivitas dengan *threshold lenient* (3000 ms) merepresentasikan kondisi yang terlalu permisif, di mana hanya keterlambatan ekstrem yang dikategorikan sebagai kegagalan sistem. Pada *threshold* ini, definisi gagal AD menjadi *total_time_timing - chunk_duration_ms > 3000*, memungkinkan toleransi delay hingga 3 detik di atas durasi konten.

Tabel 4.5. Pola Konsistensi *Acceptable Delay* Per-Pengguna. TGAD = Tidak pernah gagal AD; KGAD=Kadang-kadang gagal AD; SGAD=Selalu gagal AD

Konfigurasi	TGAD (%)	KGAD (%)	SGAD (%)
Direct No-CDN	86,4	12,8	0,9
Direct CDN	98,3	1,7	0,0
HLS No-CDN	87,2	11,9	0,9
HLS CDN	97,4	2,6	0,0

Tabel 4.6. Statistik Deskriptif Tingkat Kegagalan AD Per-Pengguna.

Konfigurasi	Rata-Rata Gagal AD (%)	Median Gagal AD (%)
Direct No-CDN	7,2	0,0
Direct CDN	0,6	0,0
HLS No-CDN	5,7	0,0
HLS CDN	1,3	0,0

Temuan pada *threshold* 3000 ms mengungkapkan bahwa hampir seluruh konfigurasi mencapai reliabilitas yang sangat tinggi, dengan kategori SGAD mendekati 0% pada seluruh protokol dan median gagal AD rate 0% untuk keempat konfigurasi. Direct CDN mencapai 98,3% pengguna yang TGAD, sedangkan HLS CDN mencapai 97,4% — perbedaan yang tidak signifikan secara praktis. Bahkan tanpa CDN, kedua protokol menunjukkan performa yang sebanding (Direct No-CDN 86,4% vs HLS No-CDN 87,2%).

Uji *Chi-square* menunjukkan bahwa distribusi gagal AD masih secara signifikan bergantung pada konfigurasi ($\chi^2 = 41,25$, $p = 2,58 \times 10^{-7}$, $df = 6$), namun efek ukuran ini jauh lebih kecil dibandingkan *threshold* 0 ms ($\chi^2 = 410,42$) dan 500 ms ($\chi^2 = 576,31$). Uji *Mann-Whitney U* antara Direct CDN dan HLS CDN mengkonfirmasi tidak terdapat perbedaan signifikan ($U = 27365,50$, $p = 0,503$), dengan median kedua konfigurasi sebesar 0%.

Threshold 3000 ms terbukti terlalu permisif untuk konteks *assessment audio* berdurasi 3–13 detik. Keterlambatan hingga 3 detik di atas durasi konten (setara dengan 23–100% dari total durasi audio) secara praktis mengganggu alur tes dan dapat menyebabkan kecemasan

peserta, sebagaimana diidentifikasi oleh faktor *technical anxiety* pada skala kecemasan tes online. Temuan Li et al. [23] mengkonfirmasi bahwa pengguna mulai meninggalkan video ketika *startup delay* melebihi 2 detik, menjadikan batas 3000 ms sebagai *threshold* yang tidak realistis untuk mengevaluasi reliabilitas sistem *assessment*.

4.5. Analisis Sensitivitas *Threshold*

Analisis sensitivitas dengan *threshold* 0 ms (*strict*), 500 ms (*moderate*), dan 3000 ms (*lenient*) dilakukan untuk memvalidasi *robustness* temuan utama. Tiga skenario ini merepresentasikan rentang toleransi yang berbeda: (1) kondisi ideal tanpa kompensasi jaringan, (2) kompromi praktis untuk *assessment* audio non-interaktif, dan (3) batas permisif yang melampaui kenyamanan pengguna.

Tripartit analisis ini memvalidasi *robustness* temuan utama: bahwa CDN merupakan determinan utama reliabilitas, dan perbedaan protokol hanya signifikan pada kondisi tanpa *edge caching*. *Threshold* 500 ms secara konsisten mempertahankan narasi ini di seluruh rentang toleransi, mengkonfirmasi bahwa temuan tidak merupakan artefak dari pemilihan *threshold* yang arbitrer.

5. KESIMPULAN

Berdasarkan analisis data telemetry dari 235 pengguna pada jaringan mobile Indonesia, penelitian ini menghasilkan temuan utama sebagai berikut:

- Karakteristik reliabilitas empat konfigurasi menunjukkan pola yang fundamental berbeda. Tanpa CDN, kedua protokol mengalami kegagalan sistematis — *Direct* No-CDN dengan 68,9% pengguna selalu gagal *acceptable delay* (AD) dan HLS No-CDN dengan 99,1% pada *threshold* 0 ms. Dengan CDN, reliabilitas meningkat secara drastis pada kedua protokol, dengan HLS CDN menunjukkan konsistensi yang lebih tinggi (73,2% tidak pernah gagal AD pada *threshold* 500 ms) dibandingkan *Direct* CDN (68,4%). Fenomena ini mengindikasikan bahwa *edge caching* merupakan prasyarat mutlak untuk reliabilitas pengiriman *audio assessment* pada jaringan dengan variabilitas tinggi.
- CDN mengurangi tingkat kegagalan secara dramatis dan independen dari protokol dan *threshold*. Pada *threshold* 500 ms, CDN menurunkan rata-rata gagal AD *rate* dari 81,7% menjadi 14,0% untuk *Direct*, dan dari 74,9% menjadi 21,1% untuk HLS. Reduksi ini bersifat proporsional dan konsisten di seluruh rentang *threshold* (0 ms, 500 ms, 3000 ms), memvalidasi *robustness* temuan.
- *Direct* dengan CDN menyediakan reliabilitas sebanding dengan HLS dengan CDN pada tingkat median, namun dengan karakteristik distribusi yang berbeda. Uji *Mann-Whitney U* pada *threshold* 500 ms mengkonfirmasi tidak terdapat perbedaan signifikan pada median gagal AD *rate* antara *Direct* CDN dan HLS CDN ($U = 30631,00$, $p = 0,706$, median keduanya 0%). Namun, *Direct* CDN menunjukkan variabilitas yang lebih tinggi (28,1% pengguna kadang-kadang gagal) dibandingkan HLS CDN (11,4%), mengindikasikan bahwa HLS memberikan konsistensi yang lebih tinggi bagi mayoritas pengguna meskipun dengan kompleksitas infrastruktur yang lebih besar. Temuan ini secara langsung menyanggah stereotip bahwa HLS secara inheren kurang cocok untuk konten berdurasi pendek — pada kondisi CDN, kedua protokol menyediakan reliabilitas yang kompetitif.

Secara kolektif, temuan ini menggeser paradigma pengambilan keputusan arsitektur pengiriman audio *assessment*: investasi pada *edge caching* memberikan dampak yang jauh lebih signifikan terhadap reliabilitas dibandingkan pilihan protokol. Pengembang sistem *assessment* berskala kecil tidak perlu secara dogmatis menolak HLS untuk konten berdurasi pendek; sebaliknya, prioritas utama harus pada implementasi CDN, dengan pilihan protokol menjadi keputusan sekunder berdasarkan kompleksitas, biaya, dan preferensi operasional. Meskipun HLS CDN menunjukkan proporsi TGAD yang sedikit lebih tinggi (73,2% vs 68,4%), perbedaan 4,8 persentase poin ini tidak bermakna secara praktis mengingat tidak terdapat perbedaan signifikan pada median (keduanya 0%). *Direct* CDN tetap direkomendasikan karena

kompleksitas infrastruktur yang jauh lebih rendah dengan reliabilitas sebanding.

UCAPAN TERIMA KASIH

Ucapan terima kasih diberikan kepada Universitas Amikom Purwokerto yang telah memfasilitasi program riset untuk mahasiswa serta dosen pembimbing yang telah bersedia memberikan ilmu serta waktunya untuk mengarahkan penulis dalam penelitian ini. Peneliti juga berterima kasih kepada pihak-pihak yang membantu peneliti untuk menyelesaikan penelitian ini, sehingga dapat selesai sesuai dengan yang diharapkan.

DAFTAR PUSTAKA

- [1] S. Uddin, M. Grega, and M. Leszczuk, "Evaluating HAS and Low-Latency Streaming Algorithms for Enhanced QoE," *Electronics*, vol. 14, no. 13, pp. 1–24, 2025.
- [2] B. Taraghi and M. Nguyen, "INTENSE: In-Depth Studies on Stall Events and Quality Switches and Their Impact on the Quality of Experience in HTTP Adaptive Streaming," *IEEE Access*, vol. 9, pp. 118087–118098, 2021, doi: 10.1109/ACCESS.2021.3107619.
- [3] A. A. El-saleh *et al.*, "Measurements and Analyses of 4G / 5G Mobile Broadband Networks: An Overview and a Case Study," *Wirel. Commun. Mob. Comput.*, vol. 2023, pp. 1–24, 2023, doi: 10.1155/2023/6205689.
- [4] C. Timmerer *et al.*, "HTTP Adaptive Streaming: A Review on Current Advances and Future Challenges," *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 21, no. 7, 2025, doi: 10.1145/3736306.
- [5] B. B. Zhang, T. Teixeira, and Y. Reznik, "Performance of Low-Latency DASH and HLS Streaming in Mobile Networks," *SMPTE Motion Imaging J.*, no. August, pp. 26–34, 2022.
- [6] S. Nie, T. Ruan, R. Chen, B. Song, S. Liu, and W. Wu, "ZeroCopy: file system assisted container buffer migration in cloud computing system," *CCF Trans. High Perform. Comput.*, vol. 7, no. 6, pp. 556–573, 2025, doi: 10.1007/s42514-025-00217-2.
- [7] K. Zhou and S. Lin, "Libra: Accelerating Socket I/O via Programmable Selective Data Copying," *ArXiv*, pp. 1–17, 2026.
- [8] E. Liotou, D. Xenakis, V. Georgara, G. Kourouniotis, and L. Merakos, "Cache-Enabled Adaptive Video Streaming: A QoE-Based Evaluation Study," *Futur. Internet*, vol. 15, no. 7, pp. 1–23, 2023, doi: <https://doi.org/10.3390/fi15070221>.
- [9] M. H. Bin Waheed *et al.*, "A New Efficient Architecture for Adaptive Bit-Rate Video Streaming," *Sustainability*, vol. 13, no. 8, pp. 1–20, 2021, doi: <https://doi.org/10.3390/su13084541>.
- [10] S. Susnjara and I. Smalley, "What is a VPS?," IBM Think. Accessed: Apr. 08, 2026. [Online]. Available: <https://www.ibm.com/think/topics/vps>
- [11] J. Wang and Z. Kai, "Performance Analysis and Optimization of Nginx based Web Server," *J. Phys. Conf. Ser.*, pp. 1–6, 2021, doi: 10.1088/1742-6596/1955/1/012033.
- [12] Nginx Inc., "Nginx Open Source Archive." Accessed: Apr. 08, 2026. [Online]. Available: <https://nginx.org/en/>
- [13] Z. P. Putro and R. A. Supono, "Comparison Analysis of Apache and Nginx Webserver Load Balancing on Proxmox VE in Supporting Server Performance," *Int. Res. J. Adv. Eng. Sci.*, vol. 7, no. 3, pp. 144–151, 2022.
- [14] D. Dzabic and J. Mårtensson, "HTTP Based Adaptive Bitrate Streaming Protocols in Live Surveillance Systems," Linköping University, 2018.
- [15] E. Mosul, M. Jajuli, and I. Maulana, "ANALISIS PERBANDINGAN PERFORMA DAN EFEKTIVITAS FRAMEWORK LARAVEL DAN NODE.JS DALAM PENGEMBANGAN WEB," *J. Inform. dan Tek. Elektro Terap.*, vol. 14, no. 2, pp. 298–306, 2026, doi: <http://dx.doi.org/10.23960/jitet.v14i2.9264>.
- [16] R. Fielding, M. Nottingham, and J. Reschke, "RFC 9112 HTTP/1.1," 2022. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc9112>
- [17] R. Fielding, Y. Lafon, and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Range Requests," 2014. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7233.html>
- [18] A. A. Barakabitze and R. Walshe, "Survey paper SDN and NFV for QoE-driven multimedia services delivery : The road towards 6G and beyond networks," *Comput. Networks*, vol. 214, pp. 1–25, 2022, doi: <https://doi.org/10.1016/j.comnet.2022.109133>.
- [19] C. Bharadwaj, U. Barman, and S. Saha, "Analysis of Latency and Bandwidth for Edge-Based Short Length in Edge Computing Devices for Short Ranges and Long Ranges," *Indian J. Sci. Technol.*, vol. 16, no. 8, pp. 531–539, 2023.
- [20] J. Weinreich, "Ensuring Consistent Performance of Web Applications," Johannes Kepler Universität Linz, 2025.
- [21] M. Dikmen, "Test anxiety in online exams:

- scale development and validity,” *Curr. Psychol.*, no. November, 2022, doi: 10.1007/s12144-022-04072-0.
- [22] ITU-T, “SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS,” 2003. [Online]. Available: <https://www.itu.int/rec/T-REC-G.114-200305-I/en>
- [23] Z. Li *et al.*, “Watching Videos from Everywhere: A Study of the PPTV Mobile VoD System,” in *IMC '12: Proceedings of the 2012 Internet Measurement Conferenc*, 2012, pp. 185–197. doi: <https://doi.org/10.1145/2398776.2398797>.
- [24] D. G. Elmes, “Within-subject experimental designs,” EBSCO Research Starters. Accessed: May 07, 2026. [Online]. Available: <https://www.ebsco.com/research-starters/health-and-medicine/within-subject-experimental-designs>
- [25] J. Ho and J. Min, “Causal Inference in Counterbalanced Within-Subjects Designs,” *ArXiv*, pp. 1–37, 2025.