# *Analisis Prioritas Buffer Video YouTube Menggunakan Algoritma Greedy Pada Tahap Initial Buffering*

Ahmad Wafi Idzharulhaqq - 13523131
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: awafi670@gmail.com

***Abstrak*—Streaming video menjadi salah satu aktivitas digital paling dominan saat ini, dan efisiensi dalam proses buffering sangat menentukan kenyamanan pengguna. Penelitian ini mensimulasikan strategi pemilihan segmen video YouTube yang diunduh lebih dahulu menggunakan pendekatan greedy scheduling. Setiap segmen memiliki nilai bitrate dan ukuran berbeda, dan sistem harus memilih beberapa segmen pertama untuk di-buffer terlebih dahulu. Dengan menggunakan rasio durasi per ukuran file sebagai kriteria pemilihan greedy, penelitian ini menunjukkan bahwa pemilihan segmen yang tepat dapat mengurangi waktu buffering awal secara signifikan. Hasil simulasi membuktikan bahwa algoritma greedy cukup efektif sebagai baseline dalam skenario terbatas koneksi.**

***Keywords—YouTube, Buffering, Greedy***

## I. PENDAHULUAN

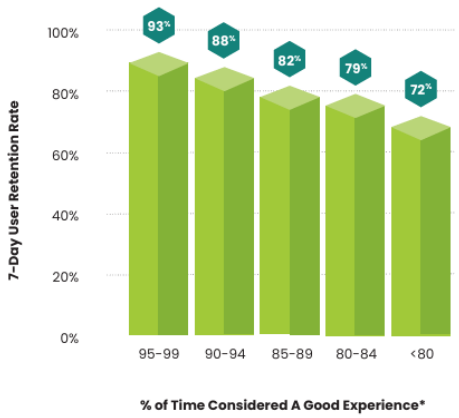
Streaming merupakan sebuah istilah untuk sebuah teknologi yang memungkinkan akses terhadap konten digital yang berada di internet. Teknologi streaming membawakan sebuah metode baru dalam menyajikan konten digital tanpa perlu melakukan pengunduhan secara penuh. Teknologi ini merupakan sebuah revolusi bagi masyarakat dalam mengonsumsi media, mengubah model distribusi dari kepemilikan fisik dalam penyimpanan, menjadi akses sesuai dengan permintaan (*on demand*). Platform streaming memungkinkan penyampaian konten atau informasi dengan cara yang lebih kontekstual dan beragam. Sebuah topik dapat dieksplorasi melalui berbagai bentuk media [1].

Akses terhadap internet yang semakin mudah membuat teknologi streaming meresap pada berbagai segmen masyarakat. Pada tahun 2024, internet mampu menjangkau lebih 5,5 juta masyarakat dunia. Jumlah ini setara dengan 68% dari total populasi yang ada di bumi [2]. Hal ini membuat teknologi streaming menjadi semakin populer diikuti oleh ketersediaan konten di dalamnya. Tidak hanya sebagai media penyebaran informasi saja, platform streaming kini digunakan sebagai media rekreasi, edukasi, hingga menjadi sumber penghasilan utama. Salah satu platform streaming yang populer di tengah-tengah masyarakat dunia saat ini adalah YouTube.

YouTube merupakan sebuah platform berbagi video di internet yang membuat pengguna dapat mengakses, mengunggah, dan membagikan video dari berbagai penjuru dunia. Youtube telah berkembang menjadi salah satu platform paling besar dan dominan di dunia sejak kemunculannya pada tahun 2005. Dengan komunitas konten kreator yang besar dan beragam, pengguna YouTube dapat mengakses berbagai jenis konten. Konten musik, *gaming*, DIY, eksperimen sosial, bahkan film dapat dinikmati oleh pengguna.

Dalam menjalankan layanannya, YouTube memiliki tantangan untuk menjaga kualitas pengalaman pengguna atau *Quality of Experience* (QoE) dengan kondisi jaringan yang dinamis. Dalam hal ini, YouTube dituntut untuk mampu memberikan kualitas penayangan video yang baik bagi setiap pengguna yang tersebar di seluruh penjuru dunia. Salah satu parameter yang paling berpengaruh terhadap QoE adalah proses *buffering*. *Buffering* adalah sebuah proses yang menunggu sementara di sisi pengguna ketika video belum siap untuk diputar. Peristiwa *buffering* seringkali terjadi karena kualitas internet yang buruk, kepadatan jaringan, performa perangkat yang digunakan, hingga masalah pada server [3].

Peningkatan kualitas video tanpa disertai dengan kemampuan koneksi internet untuk menyangga hal tersebut dapat menyebabkan buffering yang kurang menyenangkan bagi pengguna saat menikmati layanan streaming modern. Pada tahun 2022, peningkatan kualitas video yang terjadi memberikan *buffering time* sekitar 4 detik bagi pengguna Eropa, bahkan 8 detik untuk Afrika [4]. *Buffering time* sangat penting untuk diperhatikan oleh platform penyedia layanan streaming karena memiliki hubungan langsung dengan QoE pengguna. Pada sebuah riset yang dilakukan oleh Conviva di tahun 2025, pengguna akan lebih mudah kehilangan keterlibatannya apabila terdapat lebih dari 1% waktunya dalam pengalaman menggunakan layanan digital buruk. Faktanya, 2% rate dari hal ini akan menurunkan *customer engagedment* sebesar 42%. Pada kasus layanan streaming video, QoE yang buruk menyebabkan 20% pengguna tidak lagi menggunakan layanan tersebut setidaknya selama satu pekan [5].




Gambar. 1. Tingkat QoE terhadap keterlibatan pengguna pada aplikasi dalam rentang 7 hari. [5]

Dalam mengatasi hal tersebut, sistem streaming modern seperti YouTube menggunakan pendekatan streaming adaptif, seperti MPEG-DASH (Dynamic Adaptive Streaming over HTTP) dan HLS (HTTP Live Streaming).

Sebelum adaptasi kualitas dapat dilakukan secara progresif, terdapat tahapan awal berupa *initial buffering*. Pada tahap ini, sistem melakukan pengunduhan sejumlah segmen awal sebelum pemutaran video dimulai. Pada tahap ini, strategi yang digunakan dalam memilih segmen sangat penting. Jika pemutar video memilih segmen dengan *bitrate* yang besar, maka waktu dari *initial buffer* akan bertambah. Sebaliknya, jika segmen memiliki bitrate yang terlalu rendah maka kualitas gambar yang ditampilkan akan menjadi buruk dan mengurangi QoE dari pengguna.

Penelitian ini mengkaji pendekatan sederhana melalui algoritma *greedy* untuk memilih segmen awal yang paling efisien dengan rasio durasi putar tertinggi dengan ukuran file terkecil. Strategi *greedy* sangat baik dalam kasus ini dengan memanfaatkan pendekatan heuristik untuk menentukan pilihan lokal terbaik dalam setiap langkahnya [6]. Dengan menggunakan pendekatan greedy, diharapkan sistem dapat memaksimalkan pemilihan segmen untuk memaksimalkan durasi tontonan awal (bebas *buffering*) dengan pengunduhan data seminimal mungkin.

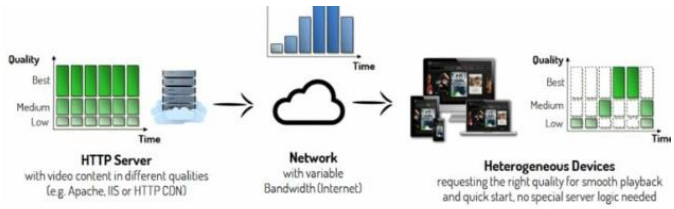
Penelitian ini fokus pada fase awal pemutaran video YouTube untuk menganalisis efektifitas dari algoritma greedy dalam menemukan solusi lokal paling efisien untuk mengurangi total data yang di unduh dan minimum *buffering time*.

## II. Landasan Teori

### A. MPEG-DASH

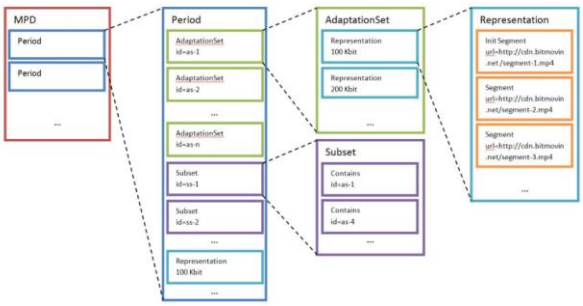
MPEG-DASH merupakan teknologi streaming independen yang memngkinkan streaming video adaptif dan disetujui sebagai standar internasional. Penggunaan MPEG-DASH memiliki tujuan untuk mengurangi *startup delay* dan *buffering* selama streaming video berlangsung. Teknologi ini juga memungkinkan adaptasi kontinu terhadap perubahan bandwith yang dinamis dalam berbagai situasi. Sebanyak 50% dari total trafik internet saat ini sudah menggunakan protokol MPEG-DASH sebagai basisnya [7].

MPEG-DASH bekerja dengan memecah media menjadi segmen-segmen yang bisa di-*encode* dengan *bitrate* yang berbeda. Segmen-segmen ini dihadirkan oleh *web server* dan dapat diakses oleh klien melalui GET HTTP standar. Selanjutja klien bertanggung jawab untuk menyesuaikan bitrate per segmen, dan memilih kualitas terbaik dari tiap segmennya berdasarkan *bandwith* dan kemampuan perangkat.




Gambar. 2. MPEG-DASH Workflow. [7]

Dalam upaya memberikan deskripsi struktur dan relasi antar segmen, MPEG-DASH menggunakan *Media Presentation Descroption* (MPD). MPD merupakan file XML yang merepresentasikan setiap segmen dan kualitas yang tersedia menggunakan HTTP *Uniform Resource Locators* (URLs). Struktur ini menyediakan pasangan segmen dengan bitrate kepada deskripsi lainnya.




Gambar. 3. MPD model. [7]

Setiap klien akan melakukan request MPD untuk mengetahui informasi terkait segmen yang tersedia, kemudian melakukan request untuk segmen individual sesuai dengan kebutuhan dan kondisinya.

### B. Adaptive Bitrate Streaming Pada YouTube

YouTube menggunakan pendekatan *Adaptive Bitrate Streaming* (ABS). Dalam pendekatan ini, video dipecah menjadi beberapa segmen kecil yang memiliki durasi tetap. Setiap segmen disediakan beberapa versi kualitas video yang berbeda. Format yang umumnya digunakan oleh YouTube adalah 144p, 360p, 480p, 720p, hingga 1080p dan lebih tinggi. Masing-masing disediakan dengan *bitrate* yang berbeda.

Pemutar video pada klien akan secara dinamis memilih kualitas berdasarkan *throughput* saat itu. Jika koneksi baik, akan dipilih versi segmen yang memiliki kualitas lebih tinggi. Sebaliknya, bila terjadi penurunan koneksi, maka versi akan dipindahkan ke kualitas yang lebih rendah. YouTube menggunakan standar dan protokol MPEG-DASH untuk melakukan ABS.

### C. *Komponen Teknis dalam Video Streaming*

Terdapat beberapa istilah yang sangat penting untuk diketahui dalam memahami bagaimana sistem video streaming bekerja. Pemahaman terhadap komponen-komponen ini menjadi fondasi utama dalam merancang strategi pengambilan segen yang efisien.

1. Segmen

Video yang disajikan melalui sistem adaptive streaming dibagi ke dalam potongan-potongan kecil yang disebut segmen. Segmen umumnya memiliki durasi tetap, seperti 2, 4, atau 6 detik. Setiap segmen berisi data video untuk rentang waktu tertentu. Pemecahan video menjadi segmen memungkinkan pemutar video untuk menyesuaikan kualitas antar segmen berdasarkan kondisi jaringan saat itu, tanpa harus memuat ulang keseluruhan video.

2. Bitrate

*Bitrate* adalah jumlah data yang dikirim per satuan waktu dalam satuan Kbps (kilobit per second). *Bitrate* berkaitan langsung dengan kualitas visual dan ukuran file. *Bitrate* yang lebih tinggi umumnya menghasilkan kualitas gambar yang lebih baik, tetapi memerlukan *bandwidth* yang lebih besar.

3. Resolusi

Setiap segmen biasanya tersedia dalam beberapa versi kualitas, masing-masing dengan resolusi dan bitrate yang berbeda. Misalnya, segmen 1 bisa tersedia dalam versi 144p (rendah), 360p (sedang), dan 720p (tinggi). Pemutar video akan memilih salah satu representasi berdasarkan kondisi jaringan saat ini.

4. Buffer

Buffer adalah memori sementara yang digunakan untuk menyimpan segmen-segmen video sebelum diputar. Tujuan utama dari buffer adalah mencegah terjadinya gangguan saat pemutaran, seperti stuttering atau freeze.

**5.** Initial Buffering

Initial buffering adalah proses awal sebelum video mulai diputar, di mana sejumlah segmen pertama diunduh dan dimasukkan ke dalam buffer. Kecepatan dan efisiensi proses ini sangat mempengaruhi startup delay, yaitu waktu tunggu sebelum video mulai berjalan. Strategi pemilihan kualitas segmen dalam fase ini menjadi fokus dari penelitian ini, karena pilihan yang buruk dapat menyebabkan waktu muat lebih lama atau kualitas video awal yang buruk.

**6**. Bandwidth

Bandwidth merujuk pada kapasitas maksimum jaringan untuk mentransfer data dalam waktu tertentu. Dalam konteks streaming, bandwidth pengguna tidak selalu stabil dan bisa berfluktuasi akibat kondisi jaringan. Sistem adaptif akan terus memantau bandwidth efektif dan menyesuaikan kualitas segmen yang diunduh untuk menjaga keseimbangan antara kualitas video dan kelancaran pemutaran.

### D. *Algoritma Greedy*

Algoritma Greedy merupakan metode yang sangat populer dan sederhana untuk permasalahan optimisasi. Algoritma ini membuat solusi optimal lokal dengan harapan bahwa solusi tersebut akan berakhir pada solusi global yang optimal [8]. Algoritma greedy menerapkan konsep "serakah" dengan memanfaatkan elemen bernilai maksimum atau nilai minimum.

Greedy tidak menjamin ditemukannya solusi optimal, terkadang algoritma greedy justru berakhir pada kemungkinan terburuk pada skala global. Namun, persoalan greedy dapat menjadi sangat kuat ketika dihadapkan pada persoalan yang memenuhi *optimal substructure*.

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← {}      { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)      { pilih sebuah kandidat dari C }
      C ← C − {x}         { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
          S ← S ∪ {x}     { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then    { solusi sudah lengkap }
    return S
  else
    write('tidak ada solusi')
  endif
```

Gambar. 4. Skema umum algoritma greedy. [9]

Dalam konteks pemilihan segmen dengan nilai efisiensi tertinggi secara lokal, algoritma greedy digunakan untuk memilih segmen yang paling efisien untuk di *buffer* terlebih dulu. Salah satu pendekatan alternatifnya adalah dengan menggunakan pemilihan segmen berdasarkan skor efisiensi rasio sebagai berikut:

$$Skor\ Efisiensi\ = \frac{Durasi\ Segmen\ (detik)}{Ukuran\ File\ (KB)}$$

Gambar. 5. Formula skor efisiensi

Pendekatan ini bersifat heuristik dan deterministik, tentang apakah suatu segmen memiliki nilai paling optimal di antara segmen lainnya, dan memilih apakah segmen tersebut akan di load ke buffer selanjutnya. Setiap segmen video dapat dianggap sebagai kandidat solusi. Pemutar video pada klien akan melakukan greedy selection terhadap segmen segmen dengan kondisi berikut:

1. Memenuhi batas maksimum kapasitas buffer
2. Memberikan total *bitrate* maksimal

## III. METODOLOGI

Penelitian ini menggunakan pendekatan simulatif dengan nmenggunakan data sintetis yang disusun dengan menyerupai karakteristik nyata sistem streaming YouTube. Fokus simulasi ini terdapat pada tahap *initial buffering*. Pada fase ini, pemutar video mengunduh sejumlah segmen awal sebelum memulai pemutaran video streaming. Pemilihan segmen awal tersebut mengimplementasikan algoritma greedy yang mempertimbangkan dua hal, yaitu efisiensi ukuran dan durasi

segmen. Program dikembangkan menggunakan bahasa Python untuk meningkatkan *readability* dan *simplicity* tanpa mengurangi fungsinya.

### A. *Parameter Dalam Dataset Simulasi*

Akan diuji 5 segmen video (5 x 3 = 15 segmen) yang digenerate secara acak melalui sebuah program, setiap segmen akan diatur untuk memiliki:

- Durasi: 4 detik per segmen [10].
- Bitrate: diatur secara acak dalam rentang 300 hingga 700 Kbps, menyesuaikan resolusi YouTube (144p – 480p) yang dipakai pada *initial buffer* [11].
- Ukuran segmen: ukuran dihitung melalui banyaknya bit yang disalurkan dalam satu detik beserta durasi pengunduhan segmen tersebut. Hasil kemudian dilakukan konversi ke dalam satuan byte sesuai formula berikut:

$$Ukuran\ (KB) = \frac{bitrate\ (Kbps) \times durasi\ (s)}{8}$$

Gambar. 6. Formula ukuran segmen

- Skor efisiensi: skor dihitung berdasarkan formula yang telah diketahui pada gambar 5. Skor ini akan menghitung berapa durasi video yang didapat per KB data. Semakin tinggi nilai skor efisien, maka kita semakin dekat dengan titik efisien.

### B. *Langkah Greedy*

Dalam penelitian ini, pendekatan Greedy digunakan untuk memilih kualitas segmen *initial buffer* secara efisien. Sistem akan menentukan satu kualitas terbaik yang perlu diambil sesuai dengan kondisi yang terjadi. Langkah-langkah algoritmanya sebagai berikut:

1. Masukan program berupa 5 segmen video, masing-masing tersedia dalam 3 kualitas bitrate yang berbeda.
2. Untuk setiap segmen, dihitung skor efisiensi kemudian dicari versi dengan skor tertinggi.
3. Hitung ukuran total data dari seluruh segmen yang sudah dipilih

### C. *Strategi Pembanding*

Untuk menguji implementasi algoritma greedy dalam proses pencarian solusi optimal untuk mengurangi *initial buffering time*, dilakukan komparasi dengan *random* dan *throughput aware selection*. Perbandingan dengan kedua algoritma ini diharapkan dapat menunjukkan tingkat efisiensi penerapan greedy di dalam permasalahan yang sedang dibahas.

## IV. Implementasi dan Pengujian

### A. *Dataset*

Dataset terdiri dari 5 segmen video yang akan memasuki *buffer* sebelum dilakukan pemutaran video. Setiap segmen akan memiliki 3 versi kualitas, yaitu rendah, sedang, dan tinggi. Masing-masing kualitas tersebut memiliki *bitrate* acak dalam kisaran sebagai berikut:

- Rendah: 300-400 Kbps
- Sedang: 401-550 Kbps
- Tinggi: 551-700 Kbps

Nilai tersebut diambil dari rentang umum *bitrate* untuk resolusi 240p hingga 480p di platform-platform streaming seperti YouTube dan layanan streaming MPEG-DASH. Pemilihan rentang ini bertujuan untuk mendekati kondisi nyata pada fase *initial buffering* video di jaringan menengah.

Untuk membuat data simulasi segmen video digital, dibuat sebuah program yang dapat menciptakan *random case* menyerupai streaming nyata.

```python
import random

def generate_segments():
    dataset = []
    for segmen_id in range(1, 6):
        versions = []
        for quality, (min_br, max_br) in zip(
            ['rendah', 'sedang', 'tinggi'],
            [(300, 400), (401, 550), (551, 700)]
        ):
            bitrate = random.randint(min_br, max_br)
            ukuran_kb = (bitrate * 4) / 8
            efisiensi = 4 / ukuran_kb
            versions.append({
                'segmen': segmen_id,
                'kualitas': quality,
                'bitrate': bitrate,
                'ukuran_kb': round(ukuran_kb, 2),
                'efisiensi': round(efisiensi, 4)
            })
        dataset.append(versions)
    return dataset
```

Gambar. 7. Program dataset.

Pada program pengembangan dataset, dilakukan sintesis data dilengkapi dengan seluruh parameter yang mensimulasikan proses segmentasi video digital di YouTube. Untuk pengujian saat ini, didapat dataset sebagai berikut:

Tabel I. Dataset Segment Video Digital

| Segmen | *Kualitas* | Parameter | | |
|---|---|---|---|---|
| | | *Bitrate (Kbps)* | *Ukuran (KB)* | *Efisiensi* |
| 1 | rendah | 335 | 167,5 | 0,0239 |
| 1 | sedang | 457 | 228,5 | 0,0175 |
| 1 | tinggi | 578 | 289,5 | 0,0138 |
| 2 | rendah | 383 | 191,5 | 0,0209 |
| 2 | sedang | 536 | 268,0 | 0,0149 |
| 2 | tinggi | 685 | 342,5 | 0,0117 |
| 3 | rendah | 388 | 194,0 | 0,0206 |
| 3 | sedang | 470 | 235,0 | 0,0170 |
| 3 | tinggi | 595 | 297,5 | 0,0134 |
| 4 | rendah | 353 | 176,5 | 0,0227 |
| 4 | sedang | 540 | 270,0 | 0,0148 |
| 4 | tinggi | 554 | 277,0 | 0,0144 |
| 5 | rendah | 380 | 190,0 | 0,0211 |
| 5 | sedang | 408 | 204,0 | 0,0196 |
| 5 | tinggi | 636 | 318,0 | 0.0126 |

Gambar. 11. Main Program

## B. *Pengujian Algoritma*

1. Algoritma Greedy

Pada implementasi algoritma greedy terhadap permasalahan ini, dilakukan pemilihan kualitaa video per segmennya dengan skor efisiensi terbaik. Implementasi greedy dapat dilihat sebagai berikut:

```
def greedy_selection(dataset):
    pilihan = []
    for segmen_versions in dataset:
        terbaik = max(segmen_versions, key=lambda x: x['efisiensi'])
        pilihan.append(terbaik)
    return pilihan
```

Gambar. 8. Program greedy dalam menentukan kualitas segmen video.

2. Random Selection

Fungsi ini mengimplementasikan strategi pemilihan acak tanpa melakukan pertimbangan apapun. Algoritma ini diharapkan dapat menjadi garis pembatas antara efisien dan tidak efisien.

```
def random_selection(dataset):
    pilihan = []
    for segmen_versions in dataset:
        acak = random.choice(segmen_versions)
        pilihan.append(acak)
    return pilihan
```

Gambar. 9. Program ramdom selection dalam menentukan kualitas segmen video.

3. Throughput Aware

Fungsi ini mengimplementasikan strategi adaptif berbasis bandwith. Untuk setiap segmen, dipilih versi dengan bitrate tertinggi namun tidak melebihi batas *bandwith*. Dalam pengujian ini, algoritma Throughput Aware atau adaptif merupakan algoritma yang paling mirip dengan algoritma yang digunakan saat ini.

```
def throughput_aware_selection(dataset, bandwidth_limit_kbps=500):
    pilihan = []
    for segmen_versions in dataset:
        cocok = [v for v in segmen_versions if v['bitrate'] <= bandwidth_limit_kbps]
        if cocok:
            terbaik = max(cocok, key=lambda x: x['bitrate'])
        else:
            terbaik = min(segmen_versions, key=lambda x: x['bitrate'])
        pilihan.append(terbaik)
    return pilihan
```

Gambar. 10. Program *adaptive/throughput aware* dalam menentukan kualitas segmen video.

Selanjutnya dilakukan pengujian dengan program sederhana untuk menemukan nilai total *initial data* yang mampu diperoleh setiap algoritma menggunakan program berikut:

```
dataset = generate_segments()
df_dataset = tampilkan_tabel_dataset(dataset)
print(df_dataset.to_string(index=False))


dataset = generate_segments()

greedy = greedy_selection(dataset)
random = random_selection(dataset)
adaptive = throughput_aware_selection(dataset, 500)

print("Total data Greedy:", total_data(greedy), "KB")
print("Total data Random:", total_data(random), "KB")
print("Total data Adaptive:", total_data(adaptive), "KB")
```

## C. *Hasil dan Analisis*

Dari percobaan berupa eksekusi terhadap ketiga strategi terhadap dataset yang sudah diperoleh, didapati hasil perbandingan total data yang dikonsumsi oleh masing-masing algoritma sebagai berikut:

| **Alogoritma** | ***Total Ukuran Data (KB)*** |
|---|---|
| Greedy | 894,5 |
| Random Selection | 1221,5 |
| Throughput-aware | 1081,0 |

Gambar. 12. Hasil pemrosesan 3 algoritma

1. Analisis Hasil Algoritma Greedy

Algoritma greedy menunjukkan hasil paling efisien dalam menggunakan data dengan total ukuran hanya 894,5 KB untuk 5 segmen awal di *initial buffer*. Melihat dari cara kerjanya, pendekatan ini mampu menghasilkan gabungan kualitas dengan data terkecil secara keseluruhan.

Keunggulan greedy terletak pada kesederhanaan logika dan efektivitas langsung yang dicapai. Tanpa memerlukan estimasi *bandwith* atau perilaku acak, algoritma greedy secara deterministik memilih versi yang paling ringan dalam konteks ukuran data, dan dapat menjadi pilihan yang ideal dalam kondisi jaringan terbatas atau sistem yang ingin meminimalkan penggunaan data. Hal ini dapat dilakukan hanya dengan mengorbankan kualitas visual dan hanya mengejar efisiensi data.

2. Analisis Hasil Algoritma Random Selection

Strategi random menghasilkan konsumsi data tertinggi yaitu 1221,5 KB. Nilai ini menunjukkan bahwa tanpa adanya dasar pengambilan keputusan yang jelas dan rasional, hasil yang didapat cenderung tidak efisien. Secara praktis, random bisa disetarakan dengan sistem pemilihan kualitas default yang tidak mempertimbangkan konteks pengguna atau jaringan. Hasil dari pemrosesan algoritma ini dapat menjadi *baseline* bahwa strategi deterministik memang lebih unggul dan dibutuhkan

3. Analisis Algoritma Throughput-aware

Strateginadaptif menghasilkan ukuran total data 1081,1 KB. Algoritma ini memiliki besar ukuran total yang berada di pertengahan dua algoritma sebelumnya. Pada penerapannya di dunia nyata, strategi ini bekerja dengan menyesuaikan pemilihan versi segmen berdasarkan kapasistas *bandwith* per segmen dan *throughput* yang dinamis. Pada akhirnya, sistem akan mempertahankan kualitas visual setinggi mungkin tanpa menimbulkan buffering berlebih.

Dari hasil yang didapat, diketahui bahwa algoritma greedy memiliki total ukuran data terendah dengan penghematan mencapai 27% dibandingkan dengan *random selection*.

## V. SIMPULAN

Penelitian ini membuktikan bahwa pendekatan greedy dapat memberikan dampak yang signifikan dalam efisiensi data dan waktu tunggu saat berada di tahap *initial buffering* sistem video live streaming, khususnya YouTube. Dibandingkan dengan algoritma *random selection* dan *throughput-aware,* greedy menunjukkan penghematan yang paling besar tanpa membutuhkan parameter jaringan tambahan.

Meskipun strategi greedy yang telah diujicobakan memiliki keterbatasan karena mengabaikan kualitas visual, pendekatan algoritma ini sangat relevan untuk tahapan *initial buffering* khususnya terhadap aplikasi yang memiliki keterbatasan *bandwith* atau memiliki kebutuhan untuk memuat video dengan cepat. Sebaliknya, strategi *throughput-aware* menyeimbangkan antara efisiensi dan stabilitas. Hal tersebut membuat algoritma *throughput-aware* atau adaptif sangat ideal untuk diterapkan pada kondisi jaringan yang dinamis

Dapat disimpulkan, bahwa algoritma greedy dapat dengan efisien diterapkan untuk tahap *initial buffering* dengan mengejar skor efisiensi tertinggi pada setiap segmennya. Penerapan ini dapat memiliki dampak yang sangat baik khususnya dalam menghindari *buffering* di tahap awal pemutaran video YouTube.

## DAFTAR PUSTAKA


[1] Wylie, Liam. “Streaming History Increasing Access to Audiovisual Archives.” *Journal of Media Practice*, vol. 7, no. 3, 16 Mar. 2007, pp. 237–248, https://doi.org/10.1386/jmpr.7.3.237_1.

[2] World Bank. “Individuals Using the Internet (% of Population) | Data.” *World Bank*, 2024, data.worldbank.org/indicator/IT.NET.USER.ZS.

[3] Bandwidth Place Team. “Buffering: What It Is & How to Stop It.” *Bandwidth Place*, 18 Oct. 2023, www.bandwidthplace.com/article/what-is-buffering-and-how-to-stop-it.

[4] Conviva. “State of Streaming Q1 2022 Report.” 2025. https://www.conviva.com.

[5] Conviva. “2025 State of Digital Experience Report Uncovering the Hidden Costs of Poor Digital Experience.” 2025. https://www.conviva.com.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, dan C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.

[7] Mueller, Christopher. “MPEG_DASH Dynamic Adaptive Streaming over HTTP | Bitmovin.” *Bitmovin*, 28 Feb. 2022, bitmovin.com/blog/dynamic-adaptive-streaming-http-mpeg-dash/. Accessed 24 June 2025.

[8] Vince, A. “A Framework for the Greedy Algorithm.” *Discrete Applied Mathematics*, vol. 121, no. 1, 15 Sept. 2002, pp. 247–260, www.sciencedirect.com/science/article/pii/S0166218X01003626, https://doi.org/10.1016/S0166-218X(01)00362-6. Accessed 1 June 2023.

[9] Munir, Rinaldi. “Algoritma Greedy (Bagian 1).” 2025.

[10] https://streaminglearningcenter.com/author/jozermindspring-com. “Choosing the Segment Duration for DASH or HLS - Streaming Learning Center.” *Streaming Learning Center*, 4 Aug. 2016, streaminglearningcenter.com/learning/choosing-the-optimal-segment-duration.html. Accessed 24 June 2025.

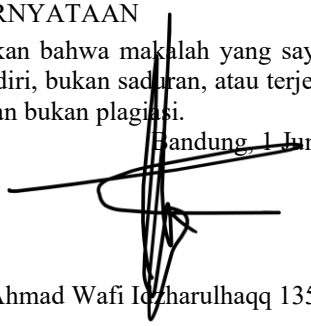
[11] Sperotto, A., Dainotti, A., & Stiller, B. (Eds.). (2020). Passive and Active Measurement. Lecture Notes in Computer Science. doi:10.1007/978-3-030-44081-7

[12] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].


## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Juni 2025



Ahmad Wafi Izzharulhaqq 13523131