PENGEMBANGAN DESAIN SISTEM *SENSITIVITY-BASED RESOURCE ALLOCATOR* UNTUK MENINGKATKAN AKURASI PADA APLIKASI *VIDEO ANALYTICS*

TUGAS AKHIR

Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Sarjana dari
Institut Teknologi Bandung

Oleh
FAISHAL ZHARFAN
NIM: 18119002
(Program Studi Teknik Telekomunikasi)

INSTITUT TEKNOLOGI BANDUNG
Mei 2024

ABSTRAK

PENGEMBANGAN DESAIN SISTEM *SENSITIVITY-BASED RESOURCE ALLOCATOR* UNTUK MENINGKATKAN AKURASI PADA APLIKASI *VIDEO ANALYTICS*

Oleh
Faishal Zharfan
NIM: 18119002
(Program Studi Teknik Telekomunikasi)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Kata kunci: pertama, kedua, ketiga.

ABSTRACT

THE DEVELOPMENT OF SENSITIVITY-BASED RESOURCE ALLOCATOR SYSTEM TO ENHANCE VIDEO ANALYTICS APPLICATIONS ACCURACY

By

Faishal Zharfan

NIM: 18119002

(*Telecommunication Engineering Program*)

LaTeXlorem ipsum (Du dkk., 2020), (Zhang dkk., 2017), (Huang dkk., 2024) *Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.*

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Keywords: first, second, third.

PENGEMBANGAN DESAIN SISTEM *SENSITIVITY-BASED RESOURCE ALLOCATOR* UNTUK MENINGKATKAN AKURASI PADA APLIKASI *VIDEO ANALYTICS*

Oleh
Faishal Zharfan
NIM: 18119002
(Program Studi Sarjana Teknik Telekomunikasi)

Institut Teknologi Bandung

Menyetujui
Tim Pembimbing

Tanggal 9 Juni 2024

Ketua

_____

Prof. Ir. Hendrawan, M.Sc., Ph.D.
NIP. 196007051987021002

PENGEMBANGAN DESAIN SISTEM *SENSITIVITY-BASED RESOURCE ALLOCATOR* UNTUK MENINGKATKAN AKURASI PADA APLIKASI *VIDEO ANALYTICS*

HALAMAN PENGESAHAN

Oleh
Faishal Zharfan
NIM: 18119002
(Program Studi Teknik Telekomunikasi)

Institut Teknologi Bandung

Menyetujui
Tim Pembimbing

Tanggal 9 Juni 2024

Ketua

_____

Prof. Ir. Hendrawan, M.Sc., Ph.D.
NIP. 196007051987021002

PEDOMAN PENGGUNAAN TUGAS AKHIR

Tugas Akhir Sarjana yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada penulis dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin penulis dan harus disertai dengan kaidah ilmiah untuk menyebutkan sumbernya.

Sitasi hasil penelitian Tugas Akhir ini dapat ditulis dalam bahasa Indonesia sebagai berikut:

Zharfan, Faishal. (2024): *Pengembangan Desain Sistem* Sensitivity-based Resource Allocator *untuk Meningkatkan Akurasi pada Aplikasi* Video Analytics, Tugas Akhir Program Sarjana, Institut Teknologi Bandung

dan dalam bahasa Inggris sebagai berikut:

Zharfan, Faishal. (2024): *The Development of Sensitivity-based Resource Allocator System to Enhance Video Analytics Applications Accuracy*, Undergraduate Final Year Project, Institut Teknologi Bandung

Memperbanyak atau menerbitkan sebagian atau seluruh Tugas Akhir haruslah seizin Dekan Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

Jangan bingung

Tidak usah repot-repot

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan karunia-Nya yang telah memberikan kesempatan penulis untuk menyelesaikan salah satu kewajiban dalam menempuh studi sarjana S1 pada Program Studi Teknik Telekomunikasi di Institut Teknologi Bandung yaitu Tugas Akhir berjudul "**Pengembangan Desain Sistem *Sensitivity-based Resource Allocator* untuk Meningkatkan Akurasi pada Aplikasi *Video Analytics***".

Ucapan terima kasih dan rasa syukur juga tidak lupa disampaikan oleh penulis kepada seluruh orang yang telah melancarkan dan membantu dalam pelaksanaan Tugas Akhir yang telah diberikan baik dalam bentuk usaha, waktu, material dan juga dukungan. Tanpa ada dukungan dari orang-orang tersebut, penulis tidak akan mampu untuk menyelesaikan pengerjaan Tugas Akhir ini dengan baik. Maka izinkanlah penulis menyampaikan rasa terima kasih kepada

1. Orang Tua penulis yang selalu memberikan dukungan finansial maupun secara moral
2. Prof. Ir. Hendrawan, M.Sc., Ph.D. selaku dosen pembimbing yang selalu membimbing dan membantu dalam pengerjaan tugas akhir ini
3. Prof. Junchen Jiang dan Prof. Haryadi S. Gunawi selaku kolaborator yang selalu memberikan masukan
4. Roy Huang selaku rekan kolaborator riset yang telah banyak membantu terkait hal teknis
5. Farhan Krishna selaku rekan TA yang selalu menjadi teman diskusi penulis

Penulisan buku tugas akhir ini tidak akan bisa dilakukan tanpa adanya orang-orang yang selalu membantu dalam penyelesaiannya. Penulis buku akhir in hanyalah manusia yang tidak lepas dari kesalahan. Maka dari itu, penulis terbuka dan menerima kritik, saran dan diskusi sebagai bahan perbaikan dan pembelajaran agar penulis dapat menjadi pribadi yang lebih baik lagi kedepannya. Semoga Buku tugas

akhir yang penulis but mampu bermanfaat bagi pembaca, terutama teman-teman pegiat telekomunikasi.

Bandung, 9 Juni 2024

Penulis

DAFTAR ISI

DAFTAR LAMPIRAN

DAFTAR GAMBAR DAN ILUSTRASI

DAFTAR TABEL

DAFTAR SINGKATAN

SINGKATAN	NAMA	KEMUNCULAN PERTAMA
API	Application Programming Interface	18
CNN	Convolutional Neural Network	5
CPU	Central Processing Unit	5
DDS	Server-Driven Video Streaming for Deep Learning Inference	6
GPU	Graphics Processing Unit	5
JCAB	Joint Configuration Adaptation and Bandwidth	5
VAP	Video Analytics Applications	2

BAB I PENDAHULUAN

1.1 Latar Belakang

Dalam 5 tahun terakhir, perkembangan teknologi telekomunikasi merupakan salah satu yang paling masif. Hal ini dibuktikan dengan kehadiran standar komunikasi mobile 5G pada tahun 2019. 5G tidak hanya dapat meningkatkan kecepatan internet menjadi 10Gb/s dari yang mulanya hanya 300Mb/s pada 4G LTE-Advanced (Khalifa, 2024a), namun 5G juga memiliki 2 *use-case* lainnya yakni *Massive Machine-Type Communication* dan *Ultra-Reliable Low-Latency Communication* (Reply, 2024).

Hadirnya *Massive Machine-Type Communication* memungkinkan banyak perangkat yang dapat terkoneksi dalam sebuah jaringan. Hal ini dapat mendukung keberlangsungan sistem IoT atau *Internet of Things*. IoT adalah sebuah teknologi komunikasi yang memungkinkan perangkat-perangkat dalam sebuah jaringan dapat berkomunikasi satu sama lainnya sehingga menciptakan sebuah sistem cerdas (Al-Fuqaha dkk., 2015). Perangkat-perangkat yang dimaksud dalam kasus ini adalah berbagai macam sensor. Data yang dikumpulkan dari berbagai macam sensor tersebut akan diproses oleh sebuah perangkat komputasi terpusat yang saat ini dikenal dengan istilah *cloud computing*

Cloud computing memiliki beberapa keterbatasan yakni lokasi yang terbatas sehingga memiliki waktu respon yang cukup lama dan tidak cocok diterapkan pada sistem yang real-time. Untuk mengatasi masalah ini, lahirlah sebuah istilah baru, yakni *Edge Computing* (Satyanarayanan, 2017). *Edge Computing* adalah sebuah paradigma dalam cloud computing yang bertujuan untuk mendekatkan proses komputasi pada perangkat *edge* sehingga pemrosesan dapat dilakukan dalam waktu singkat (Khalifa, 2023).

Masalah yang sering ditemukan pada *Edge Computing* adalah keterbatasan *bandwidth* dan waktu pemrosesan yang relatif lama terutama pada perangkat *edge* yang memiliki ukuran data yang relatif besar. Hal ini disebabkan oleh keterbatasan sumber daya komputasi yang dimiliki oleh *edge computing* (Mao dkk., 2017).

Contohnya adalah kamera keamanan, kamera akan mengirimkan video pada server untuk dilakukan proses deteksi objek (kendaraan, orang, objek lainnya). Untuk mengatasi masalah tersebut, maka diperlukan lah sebuah sistem cerdas yang dapat mengalokasikan *resource* secara periodik yang bergantung pada kebutuhan tiap kamera sehingga tidak ditemukan *backlog* pada sistem dan juga dapat memperoleh hasil yang maksimal (Cao dkk., 2020).

Dari perumusan masalah tersebut, pada tugas akhir ini akan dirancang sebuah sistem *resource allocator* pada yang berfokus pada *bandwidth* yang berbasis sensitivitas sehingga dapat meminimalkan trade-off antara alokasi bandwidth dan akurasi deteksi pada tiap kamera.

1.2 Rumusan Masalah

Masalah penelitian yang dirumuskan adalah apakah penggunaan *sensitivity-based resource allocator* yang diajukan dapat meningkatkan rata-rata akurasi dari sistem *video analytics applications* (Video Analytics Applications (VAP)) seperti DDS (Du dkk., 2020) dan lebih baik dari pendahulunya seperti VideoStorm (Zhang dkk., 2017)?

1.3 Tujuan dan Manfaat

Tujuan tugas akhir ini adalah untuk menghasilkan sebuah *sensitivity-based resource allocator* yang dapat meningkatkan rata-rata akurasi dari sistem *video analytics applications*

Hadirnya konsep smart city yang mengintegrasikan berbagai macam teknologi seperti kecerdasan buatan pada ekosistem perkotaan yang dapat membuat hidup lebih aman dan nyaman (Khalifa, 2019). Salah satu sistem yang sudah diterapkan adalah sistem untuk mendeteksi plat nomor kendaraan untuk kemudahan identifikasi kendaraan apabila terjadi sebuah pelanggaran. Beberapa tahun terakhir Kepolisian Indonesia menetapkan aturan warna terbaru plat nomor kendaraan menjadi putih, hal ini dilakukan karena warna putih dapat lebih mudah dideteksi oleh model object detection dibandingkan dengan warna hitam (Khalifa, 2024b). Dengan pengimplementasian *resource allocator*, kami berharap performa, yakni akurasi,

yang dihasilkan dari sistem tersebut dapat maksimal sehingga tidak terdapat pelanggar yang bebas tilang.

1.4 Lingkup Permasalahan

Pada topik *Enhancing Video Analytics Accuracy via Real-time Automated Video Compression Parameter Tuning as well as Enabling Resource Allocator*, secara garis besar terdiri dari 2 buah subsistem, yakni *video analytics applications* dan *resource allocator*. Fokus penulis adalah untuk mengembangkan dan menguji *resource allocator*. Selain itu, karena keterbatasan waktu yang diberikan dan alasan keamanan, *resource allocator* hanya dapat bekerja dalam jaringan lokal.

1.5 Asumsi-asumsi

Dalam pengerjaan tugas akhir ini terdapat asumsi-asumsi yang digunakan antara lain:

1. Protokol komunikasi yang digunakan adalah HTTP
2. Kondisi jaringan yang stabil sehingga komunikasi data dapat terus terjalin
3. *Resource allocator* hanya dapat dijalankan pada sistem operasi berbasis kernel Linux
4. Jumlah *client* atau kamera yang dapat terkoneksi adalah sebanyak 3 buah

1.6 Hipotesis

Penggunaan *resource allocator* yang diintegrasikan dengan gRPC dan *self-adaptive* VAP dapat memberikan gambaran kondisi kebutuhan bandwidth tiap kamera sehingga alokasi bandwidth ideal akan diperoleh yang berakibat pada peningkatan akurasi

1.7 Sistematika Buku Tugas Akhir

Pada bab 2 menjelaskan hal-hal terkait pengembangan dan perancangan *resource allocator* yang dimulai dari tinjauan pustaka, persyaratan desain, konsep desain, dan pengembangan desain.

Pada bab 3 menjelaskan hal-hal terkait model desain dan implementasi dari *resource allocator*.

Pada bab 4 menjelaskan hal-hal terkait pengujian desain dan analisis hasil pengujian dari *resource allocator* yang telah dibuat.

Pada bab 5 menjelaskan kesimpulan dari seluruh pengerjaan tugas akhir dan saran untuk pengembangan topik kedepannya.

BAB II PROSES DAN PENGEMBANGAN DESAIN

2.1 Tinjauan Pustaka

2.1.1 Resource Allocator

Kehadiran sebuah *resource allocator* merupakan hal yang sangat krusial pada keberlangsungan sistem *edge computing*. Keterbatasan sumber daya komputasi merupakan motivasi utama dibalik kebutuhan sistem ini. Sumber daya komputasi harus secara cerdas dan tepat dialokasikan sesuai dengan kebutuhan perangkat *edge* sehingga tidak terdapat *backlog* yang dapat menurukan performansi (Cao dkk., 2020). Sumber daya komputasi yang bisa dialokasikan berupa Central Processing Unit (CPU), Graphics Processing Unit (GPU), *bandwidth*, *memory*, dan *storage* (Mao dkk., 2017).

Beberapa peneliti sudah mengajukan sistem *resource allocator*, diantaranya adalah VideoStorm (Zhang dkk., 2017), Joint Configuration Adaptation and Bandwidth (JCAB) (Wang dkk., 2020), dan AutoML *for VAP* (Galanopoulos dkk., 2021). JCAB melakukan alokasi berdasarkan *bandwidth* dan model Convolutional Neural Network (CNN), artinya pada suatu sistem, terdapat beberapa model yang di-*deploy* berdasarkan resolusi video masukan. Tiap kumpulan video akan didistribusikan pada model CNN yang berbeda-beda mengikuti konfigurasi video yang sudah ditentukan sebelumnya.

Sementara AutoML *for VAP* adalah sebuah *framework* AutoML yang sengaja dibuat untuk memilih konfigurasi yang teroptimasi untuk VAP dan pembuatan model CNN. VideoStorm, di lain sisi adalah sebuah *resource allocator* yang melakukan alokasi terhadap *bandwidth*. Cara kerja VideoStorm adalah pertama-tama kamera-kamera akan mengirimkan video kepada server. Lalu pada suatu periode tertentu, VideoStorm akan melakukan alokasi *bandwidth* dengan menganalisis hasil dari beberapa detik sebelumnya dan menentukan *bandwidth* yang sesuai dengan kebutuhannya.

2.1.2 Video Analytics Application

Faktor utama keberhasilan sebuah sistem *Video Analytics* selaras dengan *Video Analytics Application* (VAP) yang digunakan. Semakin cerdas VAP maka akan semakin baik performansi yang dihasilkan oleh sistem. (Ananthanarayanan dkk., 2017) mengatakan bahwa VAP merupakan sistem yang sangat kompleks jika ingin diterapkan pada jaringan *edge*. Hal ini diakibatkan oleh video yang memiliki ukuran data yang relatif besar dan waktu pemrosesan yang cukup lama. Sehingga diperlukan pertimbangan yang sangat matang dalam mendesain sebuah VAP.

Sebuah sistem VAP terdiri dari *client*, *middleware*, dan *server*. *Client* biasanya berupa kamera yang digunakan untuk menangkap video dan melakukan beberapa proses seperti *encoding* untuk dikirimkan kepada *server*. *Middleware* adalah sebuah sistem yang menghubungkan *client* dengan *server* yang berfungsi untuk menjamin keberlangsungan transmisi data antar keduanya (IBM, 2021). Sementara *server* berguna untuk memproses data seperti melakukan deteksi objek, mengklasifikasikan objek, atau melakukan kegiatan komputasi lainnya.

Beberapa VAP sudah diajukan diantaranya adalah Server-Driven Video Streaming for Deep Learning Inference (DDS) (Du dkk., 2020), AWStream (Zhang dkk., 2018), Reducto (Li dkk., 2020), dan Glimpse (Chen dkk., 2015). DDS adalah sebuah VAP yang bekerja dengan cara mengirim video menggunakan 2 kali iterasi sehingga dapat diperoleh hasil yang baik. Salah satu persyaratan dalam mendesain VAP adalah *self-adaptability* (Jiang dkk., 2018), hal ini bermakna bahwa VAP dapat beradaptasi dengan kondisi lingkungannya dengan cara mengubah konfigurasi videonya.

2.1.3 Traffic Control

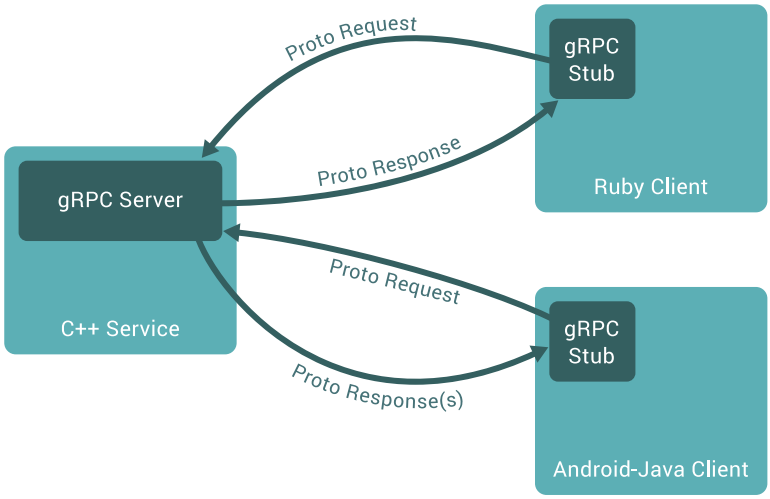
Traffic control atau lebih dikenal dengan tc adalah sebuah *tools* pada linux yang dapat digunakan untuk memegang kendali atas trafik pada kernel linux (The Linux Documentation Project, 2024). Beberapa fungsi utama dari tc adalah *SHAPING*, *SCHEDULING*, *POLICING*, dan *DROPPING*.

Salah satu *use-case* yang paling sering digunakan yakni tc akan digunakan sebagai alat yang dapat mengontrol transmisi dari sebuah *interface* jaringan, salah satunya

adalah untuk menurunkan atau mengelompokan *ingress bandwidth* ke beberapa *virtual interface* sehingga kelakuannya dapat dikontrol.

2.1.4 gRPC

gRPC adalah sebuah *framework* RPC atau *Remote Procedure Call open source* universal yang dikembangkan oleh Google (gRPC, 2024). gRPC digunakan sebagai *tools* penghubung antar perangkat atau perangkat dengan *server* yang dapat dicopot pasang secara mudah.



Gambar II-1. Use-Case gRPC (gRPC, 2024)

gRPC adalah *framework* RPC yang universal, hal ini bermakna bahwa gRPC mendukung komunikasi data menggunakan bahasa pemrograman yang berbeda seperti yang ditunjukan pada Gambar II-1. gRPC menggunakan *Protocol Buffers* atau protobuf sebagai struktur data pada pesan yang dikirimkan, secara sekilas, protobuf mirip seperti JSON, namun protobuf universal, lebih ringan, dan lebih cepat dari segi performansi.

gRPC mendukung berbagai macam jenis komunikasi yakni

1. *Unary RPC* yaitu *client* akan mengirimkan sebuah *request* dan *server* akan mengirimkan sebuah respon. Pada komunikasi ini, hanya *server* yang dapat mengirimkan respon, tidak berlaku sebaliknya

2. *Server streaming RPC* yaitu *client* akan mengirimkan sebuah *request* dan *server* akan mengirimkan respon yang tak henti hingga pesannya selesai.

3. *Client streaming RPC* yaitu kebalikannya dari *Server streaming RPC*, yakni *server* yang akan melakukan *request* dan *client* yang akan mengirimkan pesan tak henti.

4. *Bidirectional streaming RPC* yaitu *client* dan *server* dapat mengirim *request* dan mendapat respon dari satu sama lain

2.2 Persyaratan Desain

Resource allocator yang dirancang akan diintegrasikan dengan sebuah VAP terbaik dari segi performansi yakni DDS. Dengan berbagai macam pertimbangan, sistem yang didesain harus bisa memenuhi rancangan mengikuti objektif sebagai berikut

1. *Resource allocator* bersifat real-time (*Real-time design*)

2. *Resource allocator* harus bisa melakukan tradeoff yang minimal antara bandwidth dan akurasi (*Optimal Trade-off*)

3. *Resource allocator* memiliki harga pengembangan yang minimal (*Minimum Costs*)

Objektif-objektif yang disebutkan masih bersifat umum. Maka dari itu, akan dilakukan penjabaran dari tiap-tiap objektif dalam bentuk pohon subobjektif seperti pada gambar II-2 berikut ini

Dari subobjektif tersebut, dibutuhkan suatu batasan agar desain memiliki sebuah acuan yang pasti dan batasan ini tidak bisa dilanggar. Batasan atau *constraint* yang ditentukan adalah seperti yang dapat ditemukan pada tabel II-1

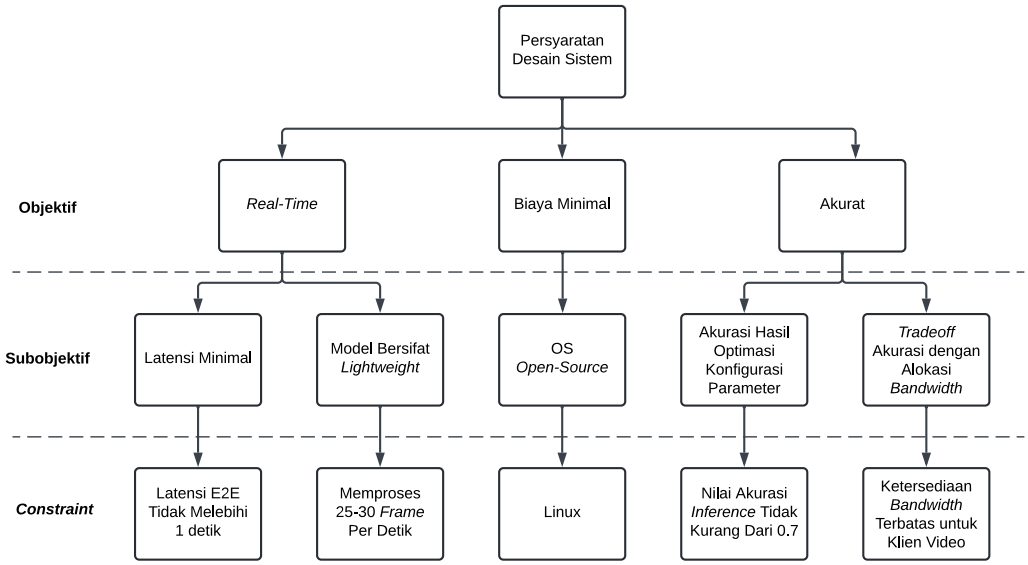
Berdasarkan Subobjektif dan *constraint* yang sudah diturunkan, diperlukan sebuah fungsi yang bertujuan untuk memenuhi persyaratan subobjektif dan *constraint*. Selanjutnya akan diturunkan persyaratan fungsional dan identifikasi fungsi-fungsi yang dipandang akan menjadi sebuah sistem seperti pada tabel II-4

Tabel II-1. Tabel *Constraint* pada Subobjektif

- **Objektif 1**: *Real-time Design*
  - **Subobjektif 1**: Latensi Minimum
    - ◎ ***Constraint***: Latensi E2E tidak melebihi 1 detik
- **Objektif 2**: *Trade-off* optimal
  - **Subobjektif 2**: Alokasi *bandwidth* terhadap akurasi *inference*
    - ◎ ***Constraint***: Ketersediaan *bandwidth* yang terbatas untuk klien sumber video
- **Objektif 3**: Harga minimum
  - **Subobjektif 3**: Sistem operasi yang bersifat *open source*
    - ◎ ***Constraint***: Harus menggunakan sistem operasi Linux

Tabel II-2. Tabel Persyaratan Fungsional dan Fungsi

Subobjektif dan *Constraint*	**Persyaratan Fungsional**	**Fungsi**
Subobjektif 1: Latensi Minimum ***Constraint***: Latensi E2E (*end-to-end*) tidak melebihi 1 detik **Subobjektif 2**: Alokasi *bandwidth* terhadap akurasi *inference* ***Constraint***: Ketersediaan *bandwidth* yang terbatas untuk klien VAP **Subobjektif 3**: Sistem operasi yang bersifat *open source* ***Constraint***: Harus menggunakan sistem operasi Linux	1. Latensi E2E (*end-to-end*) pada sistem tidak lebih dari 1 detik untuk menghindari terjadinya *backlog* saat pengukuran *overhead* 2. Kebutuhan alokasi efisien terhadap ketersediaan *bandwidth* yang terbatas untuk mencapai nilai akurasi inferensi yang terbaik 3. Linux digunakan sebagai sistem operasi karena beberapa program hanya dapat berjalan pada sistem operasi tersebut (contohnya: TC)	1. Mendesain sistem *multiprocessing* 2. Menjalankan algoritma komputasi 3. Menggunakan Linux beserta *tools*-nya



Gambar II-2. Pohon Objektif

2.3 Konsep Desain

Metrik Objektif

Pengukuran Constraint

2.4 Pengembangan Desain

Pada hakikatnya, *resource allocator* adalah sebuah perangkat lunak yang bersifat general dan ringan, artinya sistem harus dapat berjalan pada sistem manapun dan dapat melakukan komputasi secara cepat. Dengan demikian, terdapat 2 hal yang patut dipertimbangkan dalam pemilihan desain sistem, yakni bahasa pemrograman dan algoritma komputasi. Pemilihan 2 hal tersebut harus dapat memenuhi syarat fungsional yang sebelumnya sudah disebutkan yakni sistem harus dapat berjalan di bawah 1 detik. Lebih lanjut mengenai pemilihan bahasa pemrograman dan algoritma komputasi akan dibahas pada bagian berikut ini.

2.4.1 Alternatif Desain Bahasa Pemrograman Subsistem *Resource Allocator*

Pemilihan bahasa pemrograman merupakan hal yang sangat krusial terhadap proses desain dan pemilihan *tools* yang digunakan. Beberapa bahasa pemrograman dibuat

Tabel II-3. Tabel Persyaratan Fungsional dan Fungsi

Subobjektif	**Metrik Pengukuran**	**Syarat Pemenuhan**
Latensi Minimum	Tidak lebih dari 1 detik	Standard ETSI TR 101 329 V2.1.1; Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON): General aspects of Quality of Service (QoS) Standard IEEE POSIX.4; Application With Real-time Application
Alokasi *bandwidth* terhadap akurasi *inference*	Dapat meningkatkan akurasi sebesar 10%	Minimum 10% (Du dkk., 2020)
Sistem operasi yang bersifat *open source*	40% biaya digunakan dari total biaya yang dianggarkan	Maksimum 40% dari total biaya

Tabel II-4. Tabel Persyaratan Fungsional dan Fungsi

Subobjektif dan *Constraint*	**Cara Pengukuran**	**Syarat Pemenuhan**
Subobjektif: Alokasi *bandwidth* terhadap akurasi inferensi ***Constraint:*** Ketersediaan *bandwidth* yang terbatas untuk klien sumber video	Melakukan pengukuran penambahan akurasi menggunakan beberapa DDS pada bandwidth yang sama	Minimum 10% (referensi: https://dl.acm.org/doi/pdf/10.1145/3387514.340

hanya untuk kasus tertentu. Setelah melakukan beberapa studi literatur, penulis memilih 2 bahasa pemrograman yang cocok digunakan pada sistem ini, Yakni Python dan JavaScript.

Alternatif desain bahasa pemrograman pertama adalah Python, python merupakan sebuah bahasa pemrograman tingkat tinggi, berorientasi objek, dan terinterpretasi (Python, 2024). Python bersifat *portable*, artinya Python dapat dijalankan pada mesin dan sistem operasi manapun. Tidak seperti bahasa lainnya, Python memiliki struktur data yang dinamis, sehingga memudahkan *developer* dalam mengembangkan aplikasi yang membutuhkan waktu pengembangan yang cepat. Python memiliki *scientific library* yang sangat beragam dan luas, dengan demikian Python banyak digunakan oleh para scientist untuk menyelesaikan berbagai macam perhitungan kompleks. Salah satu kelemahan Python adalah bahasa pemrograman ini tidak memiliki kemampuan *multithreading* yang sangat diperlukan dalam melakukan komputasi paralel. Sebagai gantinya, Python menggunakan GIL atau *Global Interpreter Lock* yang berguna dalam melakukan *scheduling* antar thread.

Alternatif desain bahasa pemrograman pertama adalah Javascript, JavaScript merupakan sebuah bahasa pemrograman/*scripting* yang digunakan untuk mengimplementasikan fitur-fitur kompleks pada sebuah web page seperti peng-update-an konten, pengontrolan multimedia, penganimasian gambar, dll (MDN, 2024). Pada awalnya Javascript hanya diperuntukan untuk web page saja, namun pada saat ini Javascript dapat digunakan sebagai bahasa yang digunakan pada backend.

Sama seperti Python, JavaScript adalah bahasa pemrograman yang portabel sehingga bisa dijalankan pada sistem operasi manapun. Namun, berbeda dengan Python, JavaScript dapat melakukan *multithreading* sehingga dipandang memiliki kapasitas untuk melakukan komputasi yang kompleks (codingninjas, 2024). Beberapa kelemahan JavaScript antara lain adalah bersifat *asynchronous* sehingga alur pengkodeannya sukar untuk dipahami dan kurang mendukung untuk *scientific computing*.

2.4.2 Alternatif Desain Algoritma Komputasi Subsistem *Resource Allocator*

Sebuah sistem yang *realtime* tidak akan terwujud tanpa adanya algoritma yang efisien dan kencang. Berikut adalah beberapa alternatif algoritma komputasi untuk mengalokasikan besaran *bandwidth* yakni *offline learning* dan *online learning*.

Offline learning adalah sebuah teknik pembelajaran yang dilakukan oleh sebuah model machine learning atau decision maker yang biasa dikenal sebagai batch training. Sebelum melakukan pembelajaran, sistem akan mengumpulkan data terlebih dahulu, setelahnya model akan dilakukan training dengan menggunakan data tersebut. Dalam kasus lain, offline learning akan menghasilkan nilai weight yang seragam pada setiap decision pada data yang akan datang (qwak, 2024). Offline learning memiliki banyak kelebihan yakni waktu training yang relatif cepat dan kompleksitas yang kecil sehingga dapat memenuhi persyaratan fungsional real-time processing pada resource allocator. Selain itu, karena proses training berlangsung dengan sangat singkat, maka tidak ada resource yang perlu dikorbankan. Namun, mode pembelajaran ini tidak terlalu bersifat adaptif, artinya ketika terjadi perubahan pola video, decision maker tidak akan menyadari hal itu hingga waktu training selanjutnya. Sehingga dapat dikatakan bahwa offline learning cocok diterapkan pada jenis data yang memiliki pola yang seragam dan tidak cepat berubah.

Sementara itu, Online learning adalah teknik pembelajaran yang merupakan kebalikan dari offline learning. Pada online learning, data tidak dikumpulkan terlebih dahulu, namun data yang masuk akan di-training dengan mempertimbangkan model sebelumnya, hal ini akan terus berlangsung hingga program selesai. Alhasil besarnya weight pada tiap iterasi akan berbeda (Budu, Emmanuella, 2024). Secara umum, online learning bersifat adaptif, terkait hal sebelumnya, hal ini berarti bahwa online learning tahan terhadap data yang memiliki perubahan pola yang cepat dan beragam. Namun terdapat beberapa hal yang dikorbankan, salah satunya adalah kompleksitas dan penggunaan resource. Karena proses pembelajaran akan terjadi selama terus menerus, hal ini mengakibatkan kompleksitas algoritma yang tinggi memiliki kompleksitas yang lebih tinggi. Selain itu, proses pembelajaran tentunya akan menggunakan resource dan hal

ini mengakibatkan server perlu mendedikasikan sebagian resourcenya untuk keberlangsungan pembalajaran.

2.4.3 Pemilihan Desain Subsistem *Resource Allocator*

Pada bagian ini akan didiskusikan alternatif pemilihan desain bahasa pemrograman dan algoritma komputasi yang sesuai dengan spesifikasi *resource allocator*.

Tabel II-5. Tabel Pemilihan Bahasa Pemrograman

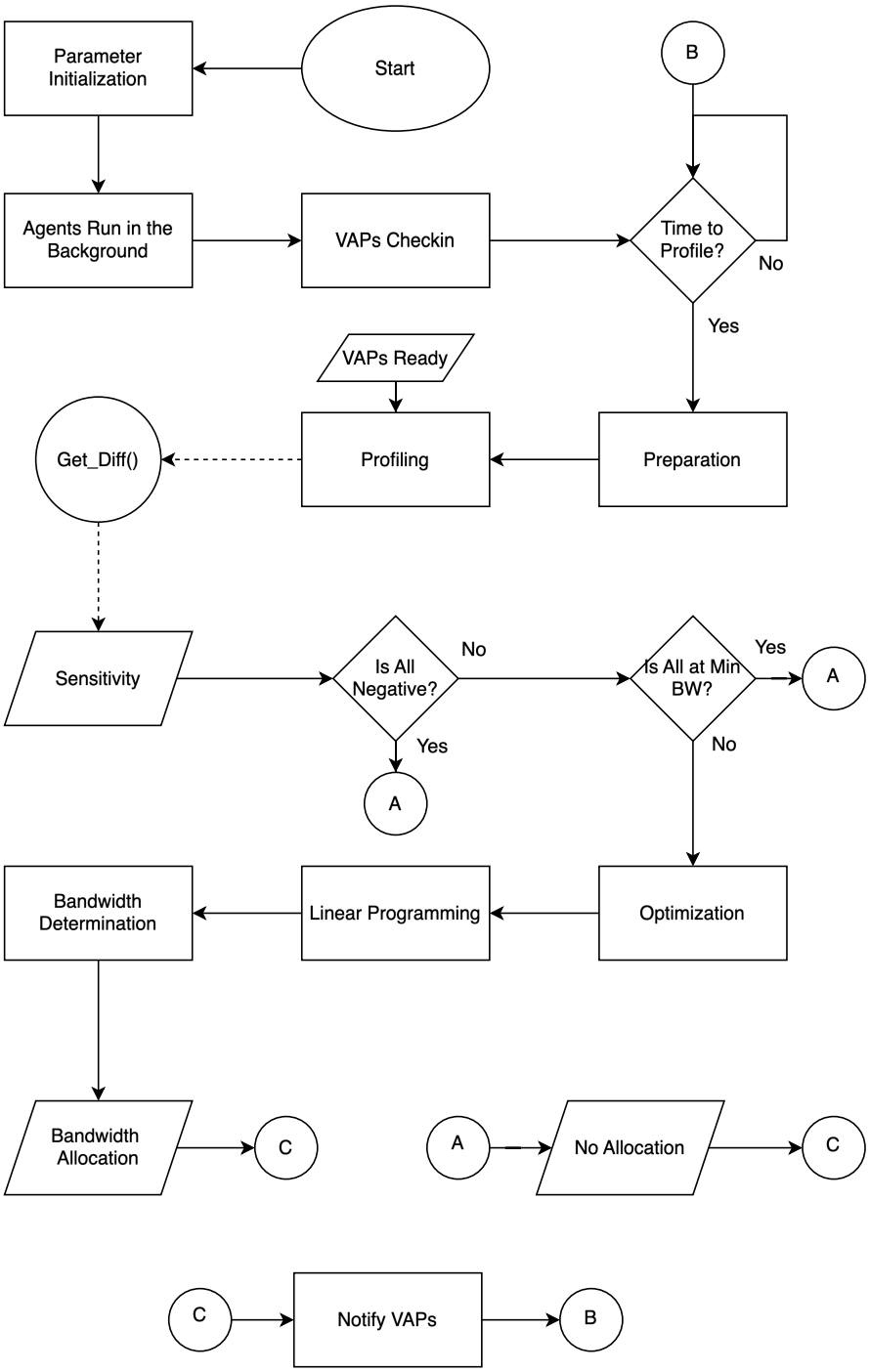
Spesifikasi	**Python**	**JavaScript**
Cross Platform	✓	✓
Easy Implementation	✓	✗
Multithreading	✗	✓
Datascience & Scientific Computing Friendly	✓	✗

Secara ringkas, perbandingan antara bahasa pemrograman Python dan JavaScript adalah seperti tertera pada tabel II-5. Python memenuhi 3 dari 4 persyaratan yang dikehendaki dan JavaScript memenuhi 2 dari 4 persyaratan yang dikehendaki. Dari hal tersebut, diketahui bahwa bahasa pemgrograman yang paling banyak memenuhi persyaratan adalah Python. Dengan demikian, bahasa pemrograman yang dipilih adalah Python karena bahasa pemrograman ini mendukung *scientific computing* dan kemudahan bahasa. Ketersediaan *scientific computing library* sangat penting karena *resource allocator* yang diajukan menggunakan algoritma *linear programming* untuk proses optimasi yang mana hal tersebut sangat krusial pada resource allocator yang menggunakan linear programming dalam pengalokasian *bandwidth* nya.

Tabel II-6. Tabel Pemilihan Algoritma Komputasi

Spesifikasi	**Online Learning**	**Offline Learning**
Computational Efficient	✗	✓
Low Complexity	✗	✓
Resource Offloading (✓ means bad)	✓	✗
Content Pattern-Independent	✓	✗

Secara ringkas, perbandingan antara Offline Learning dan Online Learning adalah seperti pada tabel II-6. Offline learning memenuhi 3 dari 4 persyaratan yang dikehendaki dan online learning hanya memenuhi 1 dari 4 persyaratan yang dikehendaki. Dari informasi tersebut, diketahui bahwa metode algoritma komputasi yang paling banyak memenuhi persyaratan adalah offline learning. Dengan demikian, alternatif desain algoritma komputasi yang dipilih dalam perancangan *resource allocator* adalah offline learning. Kompleksitas yang rendah dan tidak terdapatnya *resource offloading* sangat diperlukan keberedaannya pada *edge computing* karena keterbatasan sumber daya komputasi. Dengan demikian, *resource allocator* akan didesain menggunakan bahasa pemrograman Python dan metode algoritma komputasi offline learning.



Gambar II-3. Tingkah Laku Sistem

BAB III ANALISIS DAN PERANCANGAN

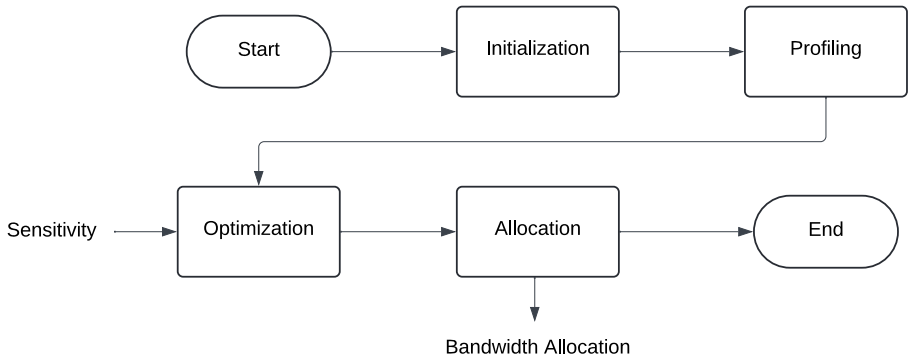
3.1 Model Desain

Secara garis besar, topik ini akan dibagi menjadi dua buah subsistem yang lebih kecil yakni *resource allocator* dan DDS. Penjelasan lebih lanjut mengenai rincian desain subsistem dan pembagian kerja akan dibahas pada tabel III-1.

Tabel III-1. Pembagian Subsistem

No	Subsistem	Rincian Desain Subsistem	Pembagian Kerja
1	*Resource Allocator*	*Resource Allocator* berperan untuk melakukan alokasi *bandwidth* secara tepat dan efisien yang bertujuan untuk meningkatkan rata-rata akurasi dari sistem VAP	Faishal Zharfan
2	DDS	DDS berperan dalam optimasi pengaturan konfigurasi parameter pengkodean dan kompresi video untuk mencapai akurasi inferensi terbaik.	Farhan Krishna

3.1.1 *Resource Allocator*



Gambar III-1. Resource Allocator Simple

Secara umum, *resource allocator* yang akan didesain memiliki alur kerja seperti pada gambar III-1 di atas. Terdapat 4 proses utama yakni *Initialization*, *Profiling*,

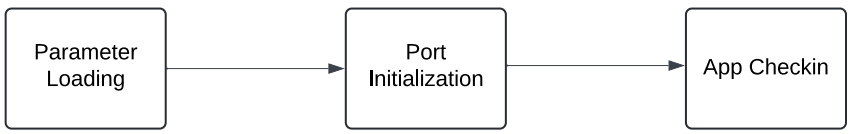
Optimization, dan *Allocation*. *Initialization* adalah sebuah proses ketika *resource allocator* pertama kali dijalankan, pada proses ini *resource allocator* akan membaca parameter-parameter yang didefinisikan, menyiapkan *port* komunikasi, dan memulai *agent* pada *background* sebagai Application Programming Interface (API). Pada proses ini pula para klien VAP akan melakukan proses *check-in* dan menjalin komunikasi dengan *resource allocator* menggunakan API gRPC, dan tahapan terakhir adalah pembagian bandwidth kepada masing-masing VAP.

Resource allocator akan berada dalam kondisi *idle* atau menunggu hingga proses *Profiling* tiba. Proses ini akan meminta para VAP yang terhubung untuk menaikan dan menurunkan *bandwidth*-nya masing-masing dan melakukan *inference* terhadap *bandwidth* tersebut. Setelahnya, hasil deteksi objek kedua *bandwidth* akan diperoleh, kedua hasil ini akan dibandingkan dengan hasil deteksi objek pada *bandwidth* saat ini, hasil ini lah yang dinamakan *inferDiff*. *inferDiff* inilah yang kemudian akan dikirim kepada *resource allocator* untuk selanjutnya diproses dan dilakukan penentuan alokasi *bandwidth*.

Setibanya *inferDiff* pada *resource allocator*, proses selanjutnya adalah *Optimization*. Pada proses ini, *inferDiff* akan diolah sedemikian sehingga tiap *inferDiff* menjadi sebuah persamaan garis yang dependen satu sama lainnya dan akan dijalankan algoritma *linear programming* untuk menentukan besaran alokasi *bandwidth* terbaik untuk tiap klien VAP. Selanjutnya hasil *Optimization* akan berupa alokasi *bandwidth* yang sesuai dengan kebutuhan VAP. *Allocation* akan mengirimkan notifikasi kepada tiap VAP terkait alokasi *bandwidth* yang mereka dapatkan. Lebih lengkap terkait implementasi proses-proses tersebut akan dibahas pada bagian selanjutnya.

3.2 Implementasi

3.2.1 Initialization



Gambar III-2. Initialization Flow Chart

Proses *Initialization* secara umum dapat dilihat pada gambar III-2. Proses pertama yang dilakukan adalah *parameter loading*, yakni membaca parameter yang terdapat pada program *runResourceAllocator.sh* seperti pada Lampiran A. Secara singkat, parameter-parameter yang digunakan adalah seperti berikut ini.

- **bandwidth** (Kbps)
- **monitorInterval** (detik)
- **bandwidthDelta** (Kb)

Parameter **bandwidth** digunakan untuk menentukan seberapa banyak *bandwidth* total yang ingin digunakan dalam sistem, dalam satuan Kbps. Sementara parameter **monitorInterval** menandakan periode *profiling* atau pengalokasian bandwidth, dapat dikatakan bahwa **monitorInterval** mengatur berapa lama jarak antar *profiling* terjardi. Terakhir, parameter **bandwidthDelta** menandakan berapa seberapa banyak bandwidth yang ingin dialokasikan dari satu video ke video yang lain dalam satu kali iterasi *profiling*.

Algoritma 3.1 Algoritma *Port Initialization*

import *grpc, threadExec*

```
1: procedure INITSERVER(workerNumber)
2:     srv ← grpc.server(threadExec(workerNumber))
3:     srv.addPort("0.0.0.0 : 5000")
4:     srv.start()
5:     srv.waitForTermination()
6: end procedure
```

Tahapan selanjutnya adalah *port initialization* yang *pseudocode* dapat dilihat pada *pseudocode* 3.1. Setelah pembacaan parameter, selanjutnya *resource allocator* akan dijalankan. Pada tahap ini program membutuhkan variabel **workerNumber** yang berguna untuk menginformasikan seberapa banyak VAP maksimal yang bisa terhubung. satu VAP yang terhubung akan memiliki kanal komunikasi sendiri dengan *resource allocator*. *Resource allocator* akan dijalankan pada **port 5000**, sehingga ketika sebuah VAP ingin terhubung dengan *resource allocator*, mereka

dapat menghubungi *port* tersebut. Selanjutnya servis dari *resource allocator* akan dihidupkan hingga terdapat *termination*.

Algoritma 3.2 Algoritma *Client Check-in*

```
1:  procedure CLIENTCHECKIN(clentRequest)
2:      NumApps ← NumApps + 1
3:      bandwidthAllocation ← MaxBW/NumApps                ▷ In Kbps
4:      app ← PipelineRepr(clientRequest)
5:      clients.add(app)
6:      for client in clients do
7:          client.limit(bandwidthAllocation)
8:          client.notify(bandwidthAllocation)
9:      end for
10: end procedure
```

Proses selanjutnya adalah *client check-in*, pada proses ini, VAP akan melakukan *check-in* kepada *resource allocator* untuk mendapatkan alokasi *bandwidth*. Selain itu, VAP yang sudah terhubung sebelumnya akan mendapat penyesuaian jumlah *bandwidth*, hal ini akan terus berlangsung hingga tidak ada VAP yang melakukan *check-in* lagi. Pada kondisi ini, baik *resource allocator* maupun VAP sudah saling terhubung dan dapat berkomunikasi satu sama lain. Perhatikan bahwa *resource allocator* memiliki fungsi *limit()*, fungsi ini diimplementasikan menggunakan TC dan berguna untuk melimitasi *data rate* yang masuk kepada *server*. Sehingga setiap VAP memiliki *inbound interface*-nya masing-masing pada server dan dilimitasi mengikuti ukuran *bandwidth*-nya.

3.2.2 Profiling

Gambar III-3. Initialization Flow Chart

Tahapan kedua adalah *Profiling*. Pada tahapan ini akan terjadi interaksi antara *resource allocator* dan VAP. Proses-proses yang terlibat dapat dilihat pada gambar

III-3.

Algoritma 3.3 Algoritma notify clients

```
1: procedure NOTIFYCLIENTS(bandwidthDelta)
2:     for client in clients do
3:         client.prepareProfiling(bandwidthDelta)
4:     end for
5:     waitForAllACK()
6: end procedure
```

Tahapan selanjutnya adalah *notify clients*, *pseudocode* mengenai proses ini adalah seperti tertera pada algoritma 3.3. Pada *notify clients*, *resource allocator* akan memberi notifikasi kepada VAP untuk bersiap-siap melakukan *profiling*. Notifikasi yang diberikan termasuk pemberian informasi mengenai seberapa besar *bandwidthDelta* yang diberikan. Setelah diberikan notifikasi, VAP akan menunggu hingga proses *inference* atau deteksi objek pada segmen video saat ini untuk selesai dan siap untuk proses *profiling* dengan memberi notifikasi kepada *resource allocator*.

Algoritma 3.4 Algoritma getInferdiff Client

```
procedure GETINFERDIFF
    for client in clients do
        client.getInferDiff()
    end for
end procedure
```

Hal ini berlanjut kepada proses selanjutnya yakni *get inferdiff*. Terdapat 2 *pseudocode* yang ditampilkan, yakni *pseudocode* pada *server* seperti terlihat pada 3.5 dan *client* seperti terlihat pada 3.5. Pada *pseudocode* 3.5, *server* akan men-*trigger client* untuk melakukan *profiling*. Sementara pada *client*, tahapan yang terjadi adalah sebagai berikut. Klien VAP akan menaikan *bandwidth* berdasarkan *deltaBandwidth* yang sudah ditetapkan sebelumnya. Selanjutnya VAP akan mengubah konfigurasi segmen video berdasarkan *bandwidth* tersebut. Konfigurasi video yang dimaksud adalah sebagai berikut

Algoritma 3.5 Algoritma getInferdiff Server

Output: $inferDiff$
procedure GETINFERDIFF
$currentBW \leftarrow currentBW + deltaBW$
$changeToBestConfig(currentBW)$
$highResult \leftarrow analyzeVideo()$
$highInferDiff \leftarrow evaluate(highResult)$
$currentBW \leftarrow currentBW - 2 * deltaBW$
$changeToBestConfig(currentBW)$
$lowResult \leftarrow analyzeVideo()$
$lowInferDiff \leftarrow evaluate(lowResult)$
$sendToServer(highInferDiff,\ lowInferDiff)$
end procedure

1. **Quantization Parameter (QP)**
2. **Resolution**
3. **Entropy**
4. **Directional Prediction**
5. **Partition Size**
6. **Motion Estimation**

Selanjutnya segmen video tersebut akan dianalisis atau dilakukan deteksi objek berdasarkan konfigurasi tersebut. Setelah analisis selesai dilakukan, *inferDiff* akan dihitung, perhitungan mengenai hal ini akan dibahas lebih lanjut pada 3.2.2.1. Lalu VAP akan menurunkan *bandwidth*-nya, persis seperti sebelumnya, konfigurasi video akan diubah kembali, dilakukan deteksi objek, dan *inferDiff* akan dihitung kembali. Pada akhirnya akan diperoleh 2 hasil, yakni *inferDiff* saat *bandwidth* dinaikan dan diturunkan. Kedua hasil inilah yang akan dikirim kembali kepada *resource allocator* untuk diproses lebih lanjut untuk diperoleh alokasi *bandwidth* yang optimal.

3.2.2.1 inferDiff

inferDiff adalah sebuah satuan yang digunakan untuk mengukur seberapa sensitif sebuah VAP terhadap perubahan bandwidth atau dikenal dengan *sensitivity*. Pada keadaan ideal, pengukuran *sensitivity* adalah selisih dari akurasi pada

bandwidth saat ini dan *bandwidth* saat dinaikan (*lowSensitivity*) maupun diturunkan (*highSensitivity*). Sehingga *pseudocode*-nya adalah seperti tertera pada 3.6

Algoritma 3.6 Algoritma Kalkulasi *Sensitivity*

```
1: procedure SENSITIVITYCALCULATION
2:     function CALCF1(result, gt)
3:         f1Score ← eval(result, gt)
4:         return f1Score
5:     end function
6:     highSensitivity ← calcF1(highRes, gt) − calcF1(currRes, gt)
7:     lowSensitivity ← calcF1(currRes, gt) − calcF1(lowRes, gt)
8:     Sensitivity ← (lowSensitivity, highSensitivity)
9: end procedure
```

Pada *pseudocode* 3.6 tersebut, terdapat variabel **gt** atau *ground truth*. *Ground truth* adalah sebuah hasil hasil deteksi objek pada keadaan yang sesungguhnya, maksudnya adalah ketika kita ingin menghitung akurasi atau *f1-score* dari sebuah model objek deteksi, hasil deteksi harus dibandingkan dengan video yang memiliki kualitas yang sangat baik. Hal ini mustahil untuk dilakukan karena *ground truth* tidak mungkin diketahui pada saat program berjalan.

Algoritma 3.7 Algoritma Kalkulasi inferDiff

```
1: procedure INFERDIFFCALCULATION
2:     function CALCF1(result, gt)
3:         f1Score ← eval(result, gt)
4:         return f1Score
5:     end function
6:     highinferDiff ← 1 − calcF1(currRes, highRes)
7:     lowinferDiff ← 1 − calcF1(lowRes, currRes)
8:     inferDiff ← (lowinferDiff, highinferDiff)
9: end procedure
```

Oleh karena hal itu, dirumuskanlah sebuah istilah baru, yakni *inferDiff*. Berbeda dengan *oracle*-nya yang membutuhkan *ground truth*, *inferDiff* akan memperlakukan hasil deteksi sebagai *ground truth* relatif. Sebagai contoh, **highResult** merupakan *ground truth* relatif terhadap **currentResult** dan **currentResult** sebagai *ground*

truth relatif terhadap **lowResult** sehingga perhitungannya adalah seperti tertera pada *pseudocode* 3.7 Walaupun tidak ideal, metode perhitungan *inferDiff* ini berkolerasi positif terhadap perhitungan *sensitivity* yakni ketika *ground truth*-nya diketahui. Sehingga *inferDiff* dapat digunakan pada sistem yang *real-time*.

3.2.3 Optimization

Gambar III-4. Optimization

Proses selanjutnya adalah *optimization*, pada bagian ini akan dilakukan perhitungan dan penentuan atas VAP mana saja yang berhak untuk memperoleh alokasi *bandwidth* dan VAP mana yang perlu dikurangi. Secara umum, tahapan yang terjadi adalah seperti tertera pada gambar III-4. Proses ini memerlukan masukan *inferDiff* yang telah diperoleh pada tahapan sebelumnya (3.2.2). Selanjutnya, *inferDiff* yang diperoleh akan diproses sedemikian rupa sehingga datanya dependen antara satu VAP dengan VAP lainnya. Lalu, dari hasil tersebut akan dilakukan pemrosesan lebih lanjut sehingga diperoleh pertidaksamaan yang akan dipecahkan atau diselesaikan dengan teknik *linear optimizatioin* atau dikenal dengan *linear programming* sehingga dapat ditentukan VAP mana yang membutuhkan *bandwidth* lebih dan mana yang dapat dikurangi.

Sebelum memroses data *inferDiff*, perlu ditentukan terlebih dahulu siapa yang jadi korban atau *victim*. Algoritma terkait pemilihan *victim* dapat dilihat pada *pseudocode* 3.8. Dalam hal ini, *victim* adalah sebuah VAP yang memiliki **lowinferDiff** yang paling tinggi dan **highinferDiff** yang paling rendah serta tidak berada pada *bandwidth* minimal, artinya *bandwidth*-nya bisa dikurangi. Dengan demikian, *victim* adalah sebuah VAP yang *bandwidthnya* akan dikurangi dan dialokasian kepada VAP lain

Seperti dijelaskan sebelumnya, *inferDiff* yang masuk pada *resource allocator* masih bersifat independen, sehingga diperlukan sebuah formula yang dapat mengatasi

Algoritma 3.8 Algoritma *Victim Determination*

```
1:  procedure VICTIMDETERMINATION(void)
2:      victim ← None
3:      victimHigh ← 1
4:      victimLow ← −1
5:      for client in clients do
6:          if client.lowinferDiff ≥ victimLow then
7:              if client.highinferDiff ≤ victimhigh then
8:                  if client.isNotMinBW then
9:                      victim ← client
10:                 end if
11:             end if
12:         end if
13:     end for
14: end procedure
```

Algoritma 3.9 Algoritma *Optimizing*

```
1:  procedure OPTIMIZING(void)
2:      for client in clients do
3:          client.sensitivity ← client.highinferDiff − victimLow
4:      end for
5:  end procedure
```

hal ini. *Pseudocode* 3.9 menghitung *sensitivity* yang dimiliki oleh masing-masing VAP. Filosofi dibalik perhitungan *sensitivity* ini adalah mengukur seberapa jauh peningkatan akurasi yang dihasilkan ketika VAP yang bersangkutan *bandwidth*-nya dinaikan dan VAP *victim* diturunkan.

Selanjutnya, *inferDiff* yang sudah dikonversi mejadi *sensitivity* akan dilakukan *linear programming* untuk menentukan VAP mana yang lebih membutuhkan tambahan *bandwidth* dengan menggunakan *library* dari Python yakni **scipy.optimize.linprog()** dengan *constraint* adalah *sensitivity* dengan ketersediaan *bandwidth*. Pada *pseudocode* 3.10, tiap VAP akan diberi dimasukan pada sebuah **array** yang berisi (**inferDiff**, **maxAllocatedBW**). **maxAllocatedBW** adalah sebuah variabel yang menyatakan jumlah maksimal *bandwidth* yang bisa dialokasikan pada VAP tersebut. Sehingga diperoleh lah data yang diperlukan. Setelah dilakukan linear programming, maka dihasilkanlah alokasi bandwidth yang diperlukan bagi masing-masing video.

Algoritma 3.10 Algoritma *Linear Programming*

import *scipy*

```
1:  procedure LINEARPROGRAMMING(void)
2:      optimizer ← [...]
3:      for client in clients do
4:          if client ≠ victim then
5:              optimizer.append((client.sensitivity, (0 deltaBW)))
6:          else
7:              optimizer.append((−1, (−deltaBW, 0)))
8:          end if
9:      end for
10:     allocation ← scipy.optimize.linprog(optimizer)
11: end procedure
```

3.2.4 Allocation

Gambar III-5. Optimization

Tahapan selanjutnya adalah allocation, dengan alur seperti pada gambar x di atas. Tahapan allocation membutuhkan sebuah masukan yakni alokasi bandwidth. Proses ini akan menginfokan alokasi yang diberikan kepada seluruh aplikasi. Pseudocodenya adalah sebagai berikut.

Algoritma 3.11 Algoritma notify Allocation

```
1: for client in clients do
2:     client.limit(client.allocation)
3:     client.notify(client.allocation)
4: end for
```

BAB IV EVALUASI DAN PEMBAHASAN

4.1 Tujuan Pengujian

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4.2 Skenario Pengujian

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4.3 Hasil Pengujian

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies

vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

4.4 Pembahasan

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

BAB V PENUTUP

5.1 Kesimpulan

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

5.2 Saran

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

DAFTAR PUSTAKA

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376.

Ananthanarayanan, G., Bahl, P., Bodík, P., Chintalapudi, K., Philipose, M., Ravindranath, L., and Sinha, S. (2017). Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67.

Budu, Emmanuella (2024). Online learning vs. offline learning.

Cao, K., Liu, Y., Meng, G., and Sun, Q. (2020). An overview on edge computing research. *IEEE Access*, 8:85714–85728.

Chen, T. Y.-H., Ravindranath, L., Deng, S., Bahl, P., and Balakrishnan, H. (2015). Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 155–168, New York, NY, USA. Association for Computing Machinery.

codingninjas (2024). Advantages and disadvantages of node.js - coding ninjas.

Du, K., Pervaiz, A., Yuan, X., Chowdhery, A., Zhang, Q., Hoffmann, H., and Jiang, J. (2020). Server-driven video streaming for deep learning inference. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 557–570, New York, NY, USA. Association for Computing Machinery.

Galanopoulos, A., Ayala-Romero, J. A., Leith, D. J., and Iosifidis, G. (2021). Automl for video analytics with edge computing. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10.

gRPC (2024). grpc - a high performance, open source universal rpc framework.

Huang, Y., Zharfan, F., Hendrawan, Gunawi, H. S., and Jiang, J. (2024). Concierge: Towards accuracy-driven bandwidth allocation for video analytics applications in edge network. In *2024 IEEE International Conference on Edge Computing & Communications (EDGE 24)*. IEEE.

IBM (2021).

Jiang, J., Ananthanarayanan, G., Bodik, P., Sen, S., and Stoica, I. (2018). Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 253–266, New York, NY, USA. Association for Computing Machinery.

Khalifa, E. (2019). Smart cities: Opportunities, challenges, and security threats. *Journal of Strategic Innovation and Sustainability*, 14(3).

Khalifa, E. (2023).

Khalifa, E. (2024a).

Khalifa, E. (2024b).

Li, Y., Padmanabhan, A., Zhao, P., Wang, Y., Xu, G. H., and Netravali, R. (2020). Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, page 359–376, New York, NY, USA. Association for Computing Machinery.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358.

MDN (2024). What is javascript? - learn web development.

Python (2024). What is python?

qwak (2024). Online vs offline machine learning - what's the difference?

Reply, T. (2024). What is 5g - an introduction to the technology.

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1):30–39.

The Linux Documentation Project (2024). tc - tool for controlling traffic in linux.

Wang, C., Zhang, S., Chen, Y., Qian, Z., Wu, J., and Xiao, M. (2020). Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 257–266.

Zhang, B., Jin, X., Ratnasamy, S., Wawrzynek, J., and Lee, E. A. (2018). Awstream: adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 236–252, New York, NY, USA. Association for Computing Machinery.

Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. (2017). Live video analytics at scale with approximation and Delay-Tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392, Boston, MA. USENIX Association.

LAMPIRAN

Lampiran A Instrumen Pengujian

Lampiran B Rincian Kasus Uji