Optimalisasi Latensi pada Game Peer-to-Peer Menggunakan Algoritma Dijkstra dan Analisis Graf Terbobot

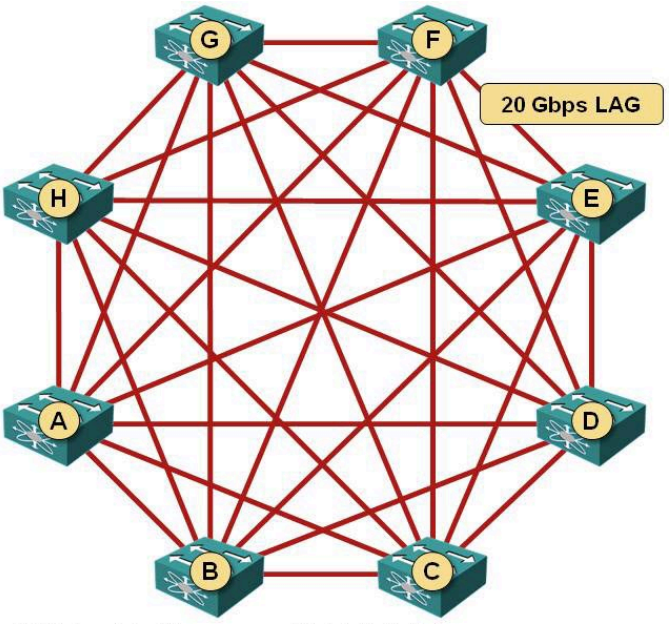
Ariel Cornelius Sitorus - 13524085
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: arielsitorus504@gmail.com , 13524085@std.stei.itb.ac.id

me online *peer-to-peer* (P2P), yang secara signifikan menurunkan kualitas pengalaman bermain. Arsitektur P2P tradisional, khususnya topologi *full-mesh*, sangat rentan terhadap "Efek Tautan Terlemah", di mana kinerja seluruh sesi game dibatasi oleh koneksi pemain paling lambat. Penelitian ini mengusulkan sebuah metodologi untuk mengoptimalkan latensi pada jaringan game P2P melalui rekonfigurasi topologi dinamis. Pendekatan ini memodelkan jaringan pemain sebagai graf terbobot, di mana setiap pemain adalah simpul, koneksi antar pemain adalah sisi, dan latensi *round-trip* adalah bobot dari sisi tersebut. Dengan menerapkan Algoritma Dijkstra, sistem dapat menemukan jalur latensi terendah antar semua pemain, tidak hanya secara langsung tetapi juga melalui pemain lain yang bertindak sebagai *relay*. Hasil dari algoritma ini digunakan untuk mengubah topologi jaringan dari *full-mesh* menjadi struktur pohon jalur terpendek (*shortest-path tree*) yang optimal. Konversi ini secara efektif menghindari koneksi berlatensi tinggi dan memanfaatkan pemain dengan konektivitas superior sebagai hub komunikasi, sehingga meminimalkan latensi maksimum dalam jaringan dan mengatasi "Efek Tautan Terlemah". Penelitian ini juga membahas tantangan implementasi praktis seperti *NAT Traversal* dan sifat dinamis jaringan internet, serta memberikan rekomendasi arsitektur berdasarkan genre game.

Keywords—Latensi, Peer-to-Peer (P2P), Algoritma Dijkstra, Graf Terbobot, Optimisasi Jaringan, Game Online, NAT Traversal

I. Pendahuluan

Arsitektur jaringan telah menjadi tulang punggung dari pengalaman game *multiplayer* modern. Dua paradigma dominan, *peer-to-peer* (P2P) dan *client-server* (C/S), menawarkan trade-off yang berbeda dalam hal biaya, skalabilitas, dan yang paling penting, latensi. Arsitektur P2P, di mana pemain terhubung langsung satu sama lain dalam topologi *full-mesh* (Gambar 1a), secara teoretis menawarkan latensi minimal karena tidak adanya perantara. Namun, model ini menderita masalah skalabilitas *bandwidth* dan sangat rentan terhadap "Efek Tautan Terlemah", di mana satu pemain dengan koneksi internet yang buruk dapat merusak pengalaman bermain untuk semua orang. Latensi yang tinggi dan tidak konsisten ini menyebabkan desinkronisasi (di mana apa yang dilihat pemain tidak sesuai dengan kenyataan), ketidakadilan dalam permainan, dan pengalaman visual yang buruk seperti karakter yang "melompat-lompat".

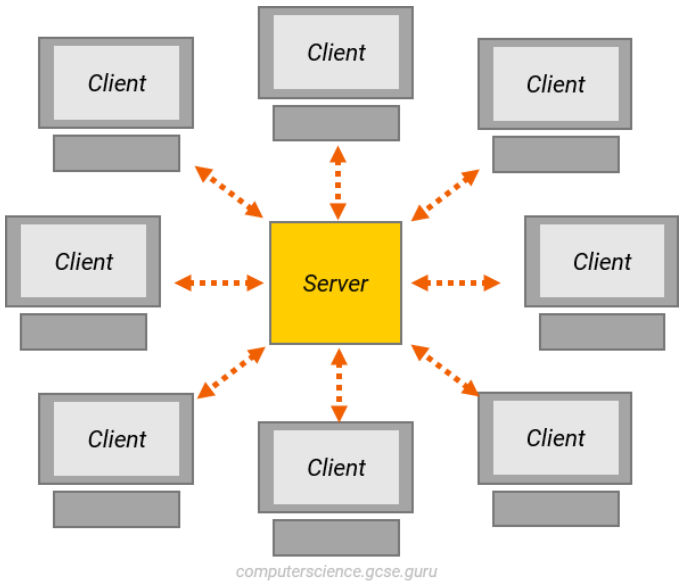


Gambar 1a.
Sumber:https://blog.ipspace.net/2012/04/full-mesh-is-worst-possible-fabric/

Di sisi lain, model C/S (Gambar 1b) menggunakan server terdedikasi sebagai otoritas pusat, memberikan pengalaman yang lebih stabil, adil, dan aman dari kecurangan, tetapi dengan biaya infrastruktur yang signifikan. Mengingat kelemahan P2P murni dan biaya tinggi C/S, ada kebutuhan untuk solusi hibrida yang dapat menggabungkan keuntungan dari kedua model tersebut.

Penelitian ini bertujuan untuk mengatasi masalah latensi dalam game P2P dengan pendekatan sistematis menggunakan teori graf dan algoritma *pathfinding*. Dengan memodelkan jaringan P2P sebagai graf terbobot, di mana latensi antar pemain menjadi bobot sisi, kita dapat mengubah masalah peningkatan kinerja game menjadi masalah optimisasi graf yang terdefinisi dengan baik. Secara spesifik, penelitian ini akan mengeksplorasi penggunaan Algoritma Dijkstra untuk

secara dinamis merekonfigurasi topologi jaringan dari *full-mesh* yang tidak efisien menjadi struktur pohon yang optimal, dengan tujuan utama untuk meminimalkan latensi pada tautan paling lambat dan dengan demikian meningkatkan kelancaran dan keadilan permainan secara keseluruhan.



Gambar1b.
Sumber:https://www.computerscience.gcse.guru/theory/client-server-networks

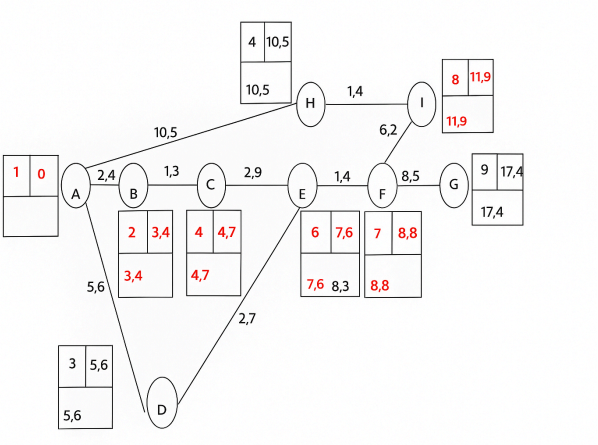
II. Landasan Teori

A. Pemodelan Jaringan P2P sebagai Graf Terbobot

Untuk menganalisis dan mengoptimalkan jaringan P2P secara sistematis, langkah pertama adalah mengabstraksikan struktur jaringan yang kompleks menjadi model matematika. Teori graf menyediakan kerangka kerja yang ideal untuk tujuan ini. Sesi game P2P dapat dimodelkan secara formal sebagai sebuah graf tak berarah (*undirected graph*) G=(V,E), di mana setiap komponen jaringan game memiliki padanan matematisnya.

1. **Simpul (Vertices, V):** Setiap pemain yang berpartisipasi dalam sesi game direpresentasikan sebagai sebuah simpul atau *node* dalam graf. Jika ada N pemain, maka graf akan memiliki N simpul.
2. **Sisi (Edges, E):** Setiap kemungkinan koneksi komunikasi langsung antara dua pemain direpresentasikan sebagai sebuah sisi atau *edge* yang menghubungkan dua simpul yang sesuai. Dalam topologi *full-mesh* P2P, setiap simpul terhubung ke semua simpul lainnya, sehingga membentuk graf lengkap (*complete graph*).
3. **Bobot (Weights, w):** Atribut paling penting dalam model ini adalah bobot yang diberikan pada setiap sisi. Bobot ini merepresentasikan "biaya" komunikasi antar *peer*. Dalam konteks optimisasi latensi, bobot sisi yang menghubungkan dua simpul adalah nilai latensi *round-trip time* (RTT) yang diukur secara dinamis antara kedua pemain tersebut, biasanya dalam milidetik (ms).

Langkah kedua adalah dengan menambahkan bobot ke setiap sisi yang saling menghubungkan simpul,



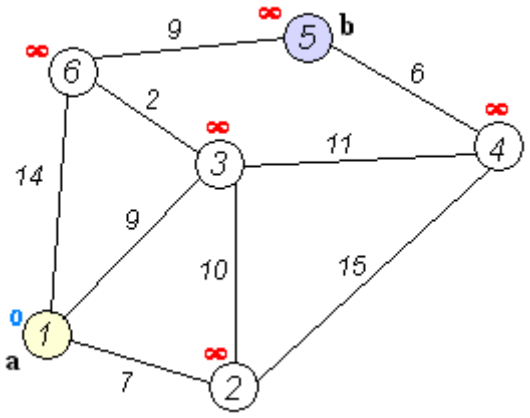
Gambar 2a.
Sumber:https://media.neliti.com/media/publications/227434-implementasi-algoritma-dijkstra-untuk-me-ecb57c78.pdf

Dengan model ini, seluruh jaringan P2P dapat direpresentasikan dalam struktur data komputasi, seperti matriks ketetanggaan (*adjacency matrix*). Representasi formal ini memungkinkan penerapan algoritma graf untuk menganalisis dan memecahkan masalah jaringan.

B. Algoritma Dijkstra

Algoritma Dijkstra adalah salah satu algoritma paling fundamental dan efisien untuk menemukan jalur terpendek dari satu simpul awal ke semua simpul lain dalam sebuah graf berbobot, dengan syarat semua bobot sisi harus non-negatif. Syarat ini terpenuhi dengan sempurna dalam konteks latensi jaringan, karena latensi tidak pernah bernilai negatif. Algoritma ini bekerja dengan pendekatan *greedy*, di mana pada setiap langkah, ia memilih jalur terpendek lokal yang tampaknya merupakan pilihan terbaik, dengan jaminan bahwa

pendekatan ini akan mengarah pada solusi optimal global.



Gambar 2b. Sumber: https://id.wikipedia.org/wiki/Algoritma_Dijkstra

Cara kerja Algoritma Dijkstra dapat diuraikan menjadi serangkaian langkah yang sistematis dan berulang (lihat Gambar 3) :

1. **Inisialisasi:** Tentukan sebuah simpul awal. Beri nilai jarak 0 pada simpul awal ini dan nilai tak terhingga (∞) untuk semua simpul lainnya. Buat sebuah daftar yang berisi semua simpul yang belum dikunjungi.
2. **Iterasi:** Selama masih ada simpul yang belum dikunjungi, ulangi langkah-langkah berikut:
    - Pilih simpul yang belum dikunjungi dengan jarak terkecil dari simpul awal. Simpul ini menjadi "simpul saat ini".
    - Untuk simpul saat ini, periksa semua simpul tetangganya yang belum dikunjungi.
    - Hitung jarak ke setiap tetangga dengan menjumlahkan jarak simpul saat ini dengan bobot sisi yang menghubungkan simpul saat ini ke tetangga tersebut.
    - Jika jarak yang baru dihitung ini lebih kecil dari jarak yang tercatat sebelumnya untuk tetangga tersebut, perbarui jaraknya. Proses ini disebut **relaksasi**.
3. **Terminasi:** Setelah semua tetangga dari simpul saat ini telah diperiksa, tandai simpul saat ini sebagai "telah dikunjungi". Simpul yang telah dikunjungi tidak akan diperiksa lagi. Algoritma berhenti ketika semua simpul telah dikunjungi.

C. NAT Traversal: STUN, TURN, dan ICE

Implementasi arsitektur P2P di dunia nyata menghadapi tantangan dari *Network Address Translation* (NAT), yang menyembunyikan alamat IP internal perangkat dan menghalangi koneksi langsung. Untuk mengatasi ini, serangkaian protokol digunakan.

1. STUN (Session Traversal Utilities for NAT): Memungkinkan sebuah *peer* untuk menemukan alamat IP publik dan jenis NAT-nya dengan menghubungi server STUN eksternal. Informasi ini digunakan untuk mencoba koneksi langsung melalui teknik *hole punching*.
2. TURN (Traversal Using Relays around NAT): Berfungsi sebagai solusi cadangan ketika koneksi langsung gagal, terutama karena NAT yang ketat (misalnya, NAT simetris). Komunikasi dialihkan melalui server TURN, yang menjamin konektivitas tetapi dengan mengorbankan peningkatan latensi karena adanya perantara.
3. ICE (Interactive Connectivity Establishment): Sebuah kerangka kerja yang mengoordinasikan STUN dan TURN. ICE mengumpulkan semua kemungkinan alamat kandidat (lokal, STUN, TURN) dan secara sistematis menguji untuk menemukan metode koneksi tercepat yang berhasil.
4.

III. Simulasi dan Implementasi

A. Metodologi Optimisasi Topologi

Inti dari solusi optimisasi terletak pada penerapan strategis Algoritma Dijkstra untuk merekonstruksi topologi jaringan dari *full-mesh* menjadi struktur pohon yang lebih efisien. Proses ini secara langsung menargetkan "Efek Tautan Terlemah".

1. **Pengukuran Latensi:** Setiap *peer* mengukur latensi ke semua *peer* lain untuk membangun matriks latensi, yang merepresentasikan graf *full-mesh* lengkap.
2. **Eksekusi Algoritma Dijkstra:** Algoritma Dijkstra dijalankan dari satu simpul *root* (misalnya, *host*) untuk membangun **pohon jalur terpendek** (*shortest-path tree*) ke semua simpul lainnya.
3. **Rekonfigurasi Topologi:** Topologi komunikasi diubah dari *full-mesh* menjadi struktur pohon ini. Data kini mengalir di sepanjang sisi-sisi pohon yang telah ditentukan, yang mungkin melibatkan pemain lain sebagai *relay node*.

Dengan metode ini, koneksi lambat dapat dihindari dan digantikan oleh jalur multi-hop yang secara agregat lebih cepat, sehingga meningkatkan kecepatan sinkronisasi untuk seluruh sesi game.

B. Contoh Simulasi

Pertama-tama, kita akan membuat algoritma Dijkstra untuk menentukan jalur terpendek dari satu simpul ke semua simpul lainnya dalam graf berbobot. Algoritma ini sangat penting dalam optimisasi rute, termasuk pada jaringan game peer-to-peer untuk meminimalkan latensi antar pemain.

```python
import heapq
import sys

def dijkstra(graph, start_node):
    distances = {node: sys.maxsize for node in graph}
    distances[start_node] = 0

    priority_queue = [(0, start_node)]

    while priority_queue:
        current_distance, current_node = heapq.heappop(priority_queue)

        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight

            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))

    return distances
```

Gambar 3a. Sumber: Arsip pengguna.

Kemudian mari kita simulasikan sesi game dengan 5 pemain menggunakan graf dari Gambar 3a. Kita akan menjalankan Algoritma Dijkstra dari simpul A.

- **Graf Awal (Latensi dalam ms):** A-B(10), A-C(50), B-C(20), B-D(40), C-D(10), C-E(60), D-E(30).
- **Langkah-langkah dari Simpul A:**
    1. **Inisialisasi:** `jarak`: {A:0, B:∞, C:∞, D:∞, E:∞}.
    2. **Iterasi 1 (Pilih A):** Perbarui jarak ke tetangga. `jarak` menjadi {A:0, B:10, C:50, D:∞, E:∞}.
    3. **Iterasi 2 (Pilih B):** Perbarui jarak ke tetangga B. Jarak ke C melalui B adalah 10+20=30, lebih baik dari 50. Jarak ke D adalah 10+40=50. `jarak` menjadi {A:0, B:10, C:30, D:50, E:∞}.
    4. **Iterasi 3 (Pilih C):** Perbarui jarak ke tetangga C. Jarak ke D melalui C adalah 30+10=40, lebih baik dari 50. Jarak ke E adalah 30+60=90. `jarak` menjadi {A:0, B:10, C:30, D:40, E:90}.
    5. **Iterasi 4 (Pilih D):** Perbarui jarak ke tetangga D. Jarak ke E melalui D adalah 40+30=70, lebih baik dari 90. `jarak` menjadi {A:0, B:10, C:30, D:40, E:70}
    6. **Iterasi 5 (Pilih E):** Selesai.

Hasil: Jalur terpendek dari A adalah: B(10), C(30 melalui B), D(40 melalui B dan C), dan E(70 melalui B, C, dan D).

Implementasi praktis harus mempertimbangkan sifat dinamis dari jaringan internet, di mana latensi terus berfluktuasi. Sebuah pohon topologi yang optimal di awal permainan bisa menjadi usang. Solusi teoretis untuk *dynamic shortest path* seringkali terlalu kompleks untuk game *real-time*. Pendekatan yang lebih pragmatis adalah melakukan pemantauan berkelanjutan pada jalur-jalur yang aktif digunakan dan memicu proses rekomputasi Algoritma Dijkstra ketika kinerja turun di bawah ambang batas tertentu. Pendekatan ini mirip dengan konsep *Resilient Overlay Networks* (RON), yang secara cerdas merutekan ulang lalu lintas untuk menghindari penurunan kinerja.

Selain itu, integrasi dengan mekanisme *NAT Traversal* seperti ICE sangat penting. Keputusan optimalisasi Dijkstra bergantung pada konektivitas yang berhasil dicapai oleh ICE. Jika jalur "optimal" ternyata memerlukan server TURN yang lambat, jalur tersebut mungkin perlu dievaluasi ulang.

Gambar 3a & gambar 3bmenunjukkan contoh implementasi sederhana dari Algoritma Dijkstra menggunakan Python, yang dapat menjadi dasar untuk modul optimisasi dalam game.

```python
graf_pemain = {
    'A': {'B': 10, 'C': 50},
    'B': {'A': 10, 'C': 20, 'D': 40},
    'C': {'A': 50, 'B': 20, 'D': 10, 'E': 60},
    'D': {'B': 40, 'C': 10, 'E': 30},
    'E': {'C': 60, 'D': 30}
}

jarak_terpendek = dijkstra(graf_pemain, 'A')
print(f"Jarak terpendek dari A: {jarak_terpendek}")
```

Gambar 3b. Sumber : Arsip pengguna.

IV. KESIMPULAN DAN SARAN

Pendekatan optimisasi topologi P2P menggunakan Algoritma Dijkstra menawarkan solusi yang kuat dan elegan untuk masalah fundamental yang melekat pada arsitektur P2P, yaitu kerentanannya terhadap "Efek Tautan Terlemah". Dengan mengubah masalah jaringan menjadi masalah optimisasi graf, metode ini secara langsung mengatasi akar penyebab latensi tinggi dan inkonsistensi dalam game P2P. Daripada membiarkan setiap pemain terhubung secara naif dalam topologi *full-mesh*, pendekatan ini secara cerdas membangun *overlay network* dalam bentuk pohon jalur terpendek.

Transformasi ini memiliki implikasi signifikan. Pertama, ia secara efektif menghilangkan ketergantungan pada koneksi pemain paling lambat, karena jalur komunikasi yang tidak efisien akan dihindari dan digantikan oleh rute multi-hop yang secara agregat lebih cepat. Kedua, metode ini secara dinamis memanfaatkan pemain dengan konektivitas superior sebagai *relay node* atau *super-peer*, mendistribusikan beban jaringan secara lebih cerdas dan efisien tanpa memerlukan infrastruktur

server terpusat yang mahal. Hal ini menciptakan arsitektur P2P yang lebih tangguh, adaptif, dan pada akhirnya memberikan pengalaman bermain yang lebih lancar dan adil bagi semua peserta.Meskipun implementasinya lebih kompleks daripada P2P murni, keuntungan dalam hal kinerja dan stabilitas latensi sangat besar, terutama untuk game dengan jumlah pemain menengah. Penelitian ini menegaskan bahwa dengan penerapan algoritma graf yang tepat, arsitektur P2P dapat berevolusi dari model yang sederhana namun rapuh menjadi sistem yang canggih dan mampu mengoptimalkan dirinya sendiri, menjadikannya pilihan yang jauh lebih layak untuk game online modern.

Ucapan terima kasih

Segala puji dan syukur dipersembahkan kepada Tuhan Yang Maha Esa atas limpahan rahmat, nikmat, dan petunjuk-Nya sehingga makalah ini dapat diselesaikan dengan baik dan tepat waktu. Terima kasih disampaikan kepada kedua dosen pengampu mata kuliah Matematika Diskrit atas ilmu, arahan, dan bimbingan yang telah diberikan selama proses perkuliahan serta dalam penyusunan makalah ini. Apresiasi yang tinggi diberikan kepada rekan-rekan seperjuangan atas semangat, kerja sama, serta diskusi yang membangun selama proses pengerjaan makalah berlangsung. Ucapan terima kasih yang tulus juga ditujukan kepada pasangan tercinta atas dukungan moral, pengertian, dan motivasi yang senantiasa menguatkan dalam setiap tahap perjalanan akademik ini.

Referensi

[1] Fitria, dan A. Triansyah, “Implementasi Algoritma Dijkstra Dalam Aplikasi Untuk Menentukan Lintasan Terpendek Jalan Darat Antar Kota Di Sumatera Bagian Selatan,” 2011.

[2] S. Rodda, M. Zichichi, G. D'Angelo, & S. Ferretti, “Modeling and Simulation of Dissemination Strategies on Temporal Graphs,” 2024.

[3] Chowdhury, F. (2020) "NAT Traversal Techniques: A Survey", International Journal of Computer Applications

[4] https://id.wikipedia.org/wiki/Algoritma_Dijkstra

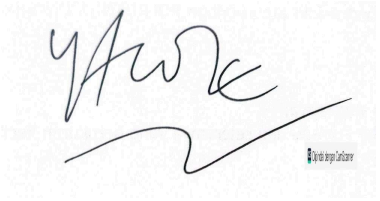
[5] Ford, B., Srisuresh, P., & Kegel, D. (2006). *Peer-to-Peer Communication Across Network Address Translators*. arXiv.

[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025-2/matdis24-25-2.htm

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



Ariel Cornelius Sitorus - 13524085