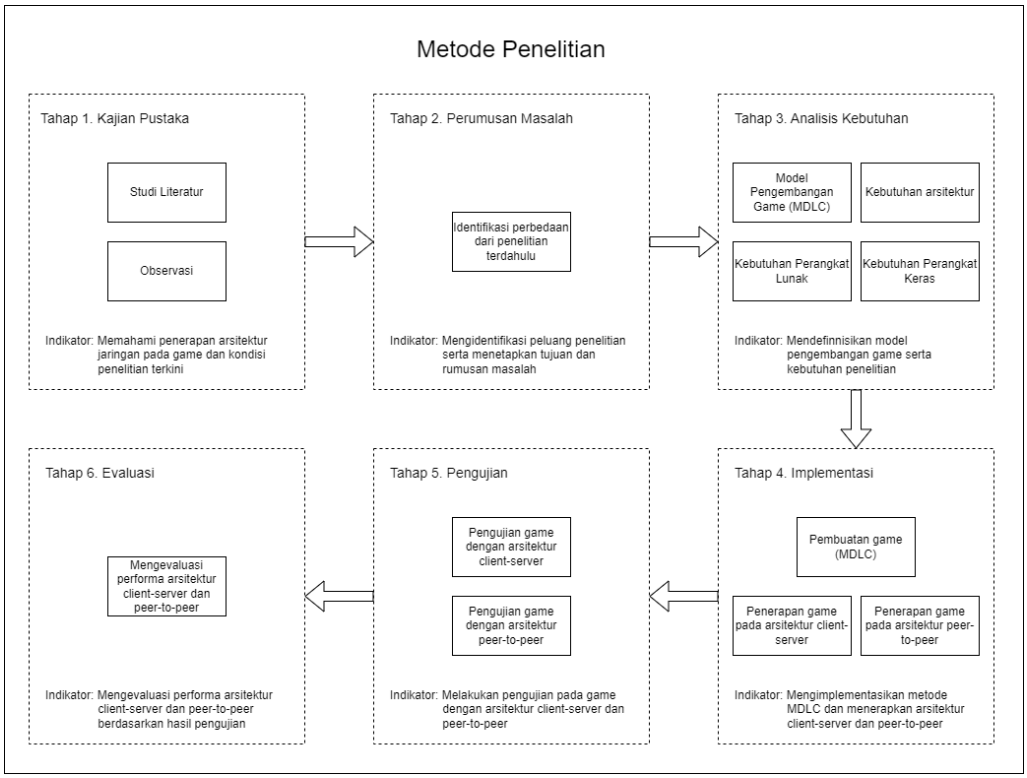
BAB III
METODOLOGI

3.1 Tahapan Penelitian

Penelitian ini menggunakan pendekatan eksperimental dengan tahapan penelitian yang disusun secara sistematis untuk mencapai tujuan penelitian. Tahapan penelitian dimulai dari kajian pustaka, perumusan masalah, analisis kebutuhan, implementasi, pengujian, hingga tahap evaluasi. Setiap tahapan saling berkaitan dan menjadi dasar bagi tahapan berikutnya sehingga hasil penelitian dapat diperoleh secara terstruktur dan dapat dipertanggungjawabkan secara ilmiah.



Gambar 3.1 Tahapan Penelitian

Tahapan penelitian ini disusun secara sistematis untuk memastikan proses pengembangan, pengujian, dan evaluasi arsitektur jaringan pada game *multiplayer*

dapat dilakukan secara terstruktur dan terukur. Alur penelitian mengacu pada tahapan yang ditunjukkan pada Gambar 3.1 yang terdiri dari enam tahap utama, yaitu kajian pustaka, perumusan masalah, analisis kebutuhan, implementasi, pengujian, dan evaluasi performa.

Tahap pertama adalah kajian pustaka, yang bertujuan untuk memperoleh pemahaman komprehensif mengenai penerapan arsitektur jaringan pada game multiplayer. Pada tahap ini dilakukan studi literatur terhadap penelitian-penelitian terdahulu yang membahas arsitektur *client-server* dan *peer-to-peer* (P2P), metode pengembangan game, serta teknik pengujian sistem. Selain itu, observasi terhadap tren dan kondisi penelitian terkini dilakukan untuk mengidentifikasi pendekatan yang umum digunakan serta celah penelitian yang masih belum banyak dikaji. Hasil dari tahap ini menjadi dasar konseptual bagi penelitian yang dilakukan.

Tahap kedua adalah perumusan masalah, yang dilakukan berdasarkan hasil kajian pustaka. Pada tahap ini diidentifikasi perbedaan, keterbatasan, dan kekurangan penelitian terdahulu, khususnya terkait kurangnya evaluasi performa kuantitatif dan minimnya perbandingan antara arsitektur client-server dan P2P pada game *multiplayer* jaringan lokal. Berdasarkan identifikasi tersebut, ditetapkan rumusan masalah dan tujuan penelitian yang berfokus pada analisis dan perbandingan performa kedua arsitektur jaringan tersebut.

Tahap ketiga adalah analisis kebutuhan, yang bertujuan untuk mendefinisikan seluruh kebutuhan penelitian sebelum tahap implementasi dilakukan. Analisis ini mencakup penentuan model pengembangan game yang

digunakan, yaitu *Multimedia Development Life Cycle* (MDLC), analisis kebutuhan arsitektur jaringan yang akan diterapkan, serta kebutuhan perangkat keras dan perangkat lunak yang diperlukan untuk pengembangan, pengujian, dan evaluasi sistem. Hasil analisis kebutuhan ini menjadi acuan teknis dalam proses implementasi agar penelitian berjalan sesuai dengan tujuan yang telah ditetapkan.

Tahap keempat adalah implementasi, yaitu tahap pembuatan game multiplayer dengan mengacu pada model MDLC yang telah ditentukan. Pada tahap ini, game yang sama diimplementasikan pada dua arsitektur jaringan yang berbeda, yaitu arsitektur *client-server* dan arsitektur *peer-to-peer* (P2P). Implementasi dilakukan dengan menjaga kesamaan mekanisme *gameplay*, logika permainan, dan fitur utama agar perbedaan hasil yang diperoleh pada tahap pengujian performa benar-benar disebabkan oleh perbedaan arsitektur jaringan, bukan oleh faktor lain. Pengujian fungsional sistem menggunakan metode *Black-Box* tetap dilaksanakan sebagai tahap validasi awal, pengujian fungsionalitas ini bertujuan untuk memastikan bahwa kedua implementasi arsitektur memiliki fungsi dan mekanisme permainan yang berjalan secara setara sebelum dilakukan proses pengambilan data performa. Dengan dilakukannya pengujian fungsional, validitas dan reliabilitas alat uji dapat lebih terjamin karena sistem yang digunakan telah dipastikan bekerja sesuai dengan kebutuhan dan rancangan yang ditetapkan, tahap ini juga mendukung aspek implementasi yang tercantum dalam judul penelitian sehingga penelitian tidak hanya berfokus pada analisis performa, tetapi juga pada keberhasilan penerapan sistem yang diuji. Pengujian *Black-Box* juga dilakukan untuk menjamin kesetaraan kondisi pengujian pada kedua arsitektur sehingga perbandingan

performa dapat dilakukan secara objektif dan seimbang. Dengan memastikan seluruh fitur utama berjalan dengan baik, potensi munculnya anomali data akibat kesalahan fungsi atau bug sistem dapat diminimalkan sehingga hasil pengukuran performa lebih akurat dan representatif. Pelaksanaan pengujian fungsional sebelum tahap evaluasi performa juga sejalan dengan standar pada metode *Multimedia Development Life Cycle* di mana validasi fungsi sistem merupakan bagian penting sebelum sistem digunakan dalam tahap pengujian lanjutan.

Tahap kelima adalah pengujian performa yang bertujuan untuk mengukur performa jaringan pada masing-masing arsitektur. Pengujian dilakukan terhadap game dengan arsitektur *client-server* dan *peer-to-peer* pada lingkungan yang sama. Tahap ini mencakup pengumpulan data yang diperlukan untuk analisis performa jaringan, sehingga dapat diperoleh hasil pengujian yang valid dan dapat dibandingkan.

Tahap terakhir adalah evaluasi performa, yaitu tahap analisis terhadap data hasil pengujian yang telah diperoleh. Evaluasi dilakukan untuk membandingkan performa arsitektur *client-server* dan *peer-to-peer* berdasarkan metrik performa yang telah ditentukan. Hasil evaluasi ini digunakan untuk menjawab rumusan masalah penelitian serta menarik kesimpulan mengenai kelebihan, kekurangan, dan kondisi penggunaan yang tepat dari masing-masing arsitektur jaringan pada game multiplayer jaringan lokal.

3.2 Analisa Kebutuhan

Analisa kebutuhan merupakan tahap penting dalam penelitian ini untuk memastikan bahwa proses implementasi dan pengujian game *multiplayer* dapat dilakukan secara sistematis, terukur, dan sesuai dengan tujuan penelitian. Tahap ini bertujuan untuk mendefinisikan seluruh kebutuhan yang berkaitan dengan pengembangan game, penerapan arsitektur jaringan, serta lingkungan pengujian yang akan digunakan dalam membandingkan performa arsitektur *client-server* dan *peer-to-peer*.

Analisa kebutuhan dilakukan setelah tahap perumusan masalah sehingga seluruh kebutuhan yang ditentukan secara langsung mendukung pencapaian tujuan penelitian, yaitu mengevaluasi dan membandingkan performa kedua arsitektur jaringan pada lingkungan game *multiplayer* yang sama. Oleh karena itu, analisa kebutuhan tidak hanya mencakup aspek pengembangan game, tetapi juga mencakup kebutuhan teknis jaringan, perangkat keras, dan perangkat lunak yang diperlukan untuk mendukung proses pengujian dan evaluasi performa secara kuantitatif.

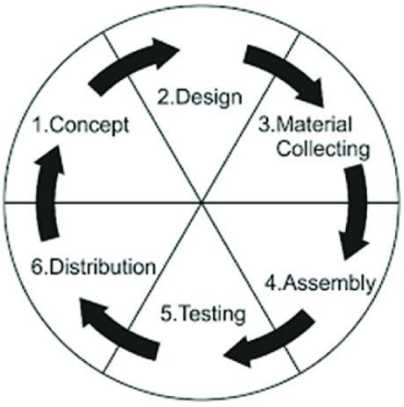
Pada penelitian ini, analisa kebutuhan dibagi menjadi beberapa komponen utama. Pertama, analisa terhadap model pengembangan game yang digunakan, yaitu *Multimedia Development Life Cycle*, untuk memastikan alur pengembangan game berjalan terstruktur dan terdokumentasi dengan baik. Kedua, analisa kebutuhan arsitektur jaringan, yang mencakup perancangan dan penerapan dua arsitektur berbeda, yaitu *client-server* dan *peer-to-peer*, pada game yang sama agar hasil perbandingan bersifat adil dan valid. Ketiga, analisa kebutuhan perangkat

keras, yang meliputi spesifikasi perangkat yang digunakan sebagai server dan *client* selama proses pengembangan dan pengujian. Keempat, analisa kebutuhan perangkat lunak, yang mencakup game engine, framework networking, sistem operasi, serta tools pendukung pengembangan dan pengujian.

Hasil dari analisa kebutuhan ini menjadi dasar teknis dalam tahap implementasi dan pengujian, sehingga perbedaan hasil performa yang diperoleh dapat dikaitkan secara langsung dengan perbedaan arsitektur jaringan yang digunakan, bukan disebabkan oleh ketidaksesuaian lingkungan atau keterbatasan perangkat. Dengan demikian, analisa kebutuhan berperan penting dalam menjamin validitas dan reliabilitas hasil penelitian.

3.2.1 Metode Pengembangan Game

Tahap pengembangan game dilakukan dengan mengacu pada metode MDLC yang terdiri dari tahapan *concept*, *design*, *material collecting*, *assembly*, *testing*, dan *distributing*.



Gambar 3.2 Tahapan Metode MDLC

Pada tahap *concept* ditentukan konsep dasar game, termasuk genre, mekanisme permainan, serta skenario *multiplayer* yang akan digunakan sebagai

objek penelitian. Konsep game dirancang sederhana agar fokus penelitian tetap pada aspek performa jaringan, bukan kompleksitas gameplay.

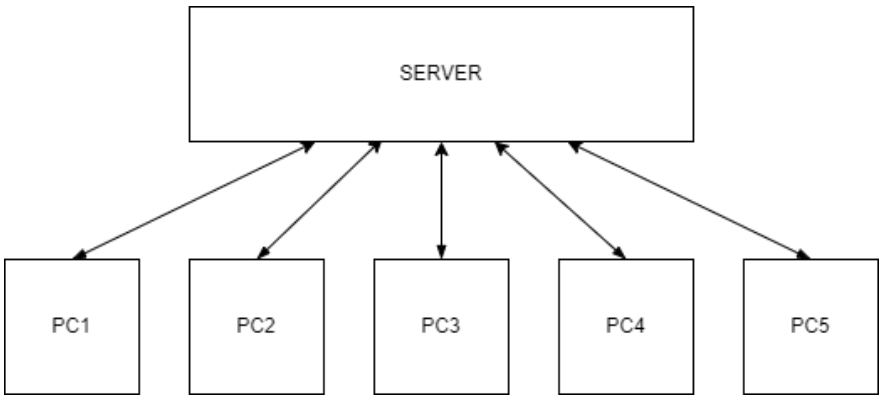
Tahap *design* mencakup perancangan sistem game, arsitektur jaringan, serta alur komunikasi data antar pemain. Pada tahap ini, dua arsitektur jaringan dikembangkan secara terpisah, yaitu *client-server* dan *peer-to-peer*, namun dengan mekanisme permainan yang sama. Selanjutnya, tahap *material collecting* dilakukan untuk mengumpulkan aset pendukung seperti grafis, audio, dan elemen antarmuka yang diperlukan dalam game.

Tahap assembly merupakan proses implementasi seluruh rancangan ke dalam game menggunakan game engine Unity. Pada tahap ini, logika permainan dan sistem *multiplayer* diimplementasikan sesuai dengan arsitektur yang telah dirancang. Setelah implementasi selesai, tahap *testing* dilakukan untuk memastikan bahwa game dapat berjalan dengan baik pada kedua arsitektur jaringan, serta memastikan tidak terdapat kesalahan fungsional sebelum dilakukan pengujian performa. Tahap *distributing* dilakukan dengan membangun aplikasi game dalam bentuk *executable* untuk kebutuhan pengujian pada perangkat yang berbeda.

3.2.2 Kebutuhan Arsitektur

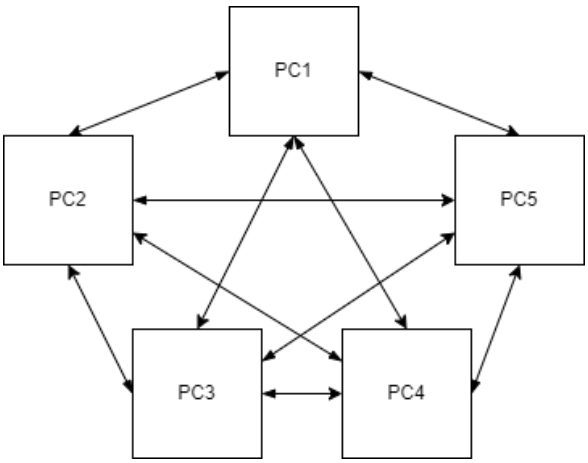
Kebutuhan arsitektur dalam penelitian ini didefinisikan untuk mendukung tujuan utama penelitian, yaitu membandingkan performa arsitektur jaringan *client-server* dan *peer-to-peer* pada game *multiplayer* berbasis jaringan lokal. Perancangan arsitektur dilakukan dengan memastikan bahwa kedua pendekatan jaringan diimplementasikan pada game yang sama, dengan mekanisme gameplay,

logika permainan, dan lingkungan pengujian yang identik, sehingga perbedaan hasil pengujian dapat merepresentasikan karakteristik arsitektur jaringan yang digunakan.



Gambar 3.3 Kebutuhan Arsitektur *Client-Server*

Pada arsitektur *client-server* dibutuhkan sebuah node yang berperan sebagai server dan memiliki otoritas penuh terhadap pengelolaan state permainan. Server bertugas menangani koneksi antar *client*, sinkronisasi data permainan, validasi aksi pemain, serta distribusi *update state* ke seluruh *client* yang terhubung. Arsitektur ini dirancang untuk meminimalkan inkonsistensi data dan konflik *state* antar pemain, dengan konsekuensi adanya ketergantungan terhadap satu titik pusat. Dalam penelitian ini, implementasi arsitektur *client-server* dirancang menggunakan framework Mirror, yang menyediakan mekanisme *authoritative server*, sinkronisasi objek, serta manajemen koneksi yang sesuai untuk pengujian pada jaringan lokal.



Gambar 3.4 Kebutuhan Arsitektur *Peer-to-peer*

Pada arsitektur *peer-to-peer* tidak terdapat server terpusat yang bertindak sebagai pengendali utama permainan. Setiap *node* atau pemain berperan sebagai *peer* yang dapat mengirim dan menerima data secara langsung dari *peer* lainnya. Kebutuhan arsitektur P2P dalam penelitian ini mencakup mekanisme komunikasi langsung antar *peer*, pengelolaan sinkronisasi *state* permainan secara terdistribusi, serta penanganan konflik data yang mungkin terjadi akibat tidak adanya otoritas tunggal. Implementasi arsitektur P2P dirancang menggunakan TCP/IP native dari NET Framework yang mendukung komunikasi reliable dengan *ordered packet delivery* dan sesuai untuk kebutuhan sinkronisasi data game pada jaringan lokal. Agar perbandingan performa bersifat objektif, kedua arsitektur harus memenuhi kebutuhan arsitektural yang setara, meliputi jumlah pemain yang sama, skenario gameplay yang identik, serta parameter pengujian yang konsisten.

3.2.3 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras dalam penelitian ini ditentukan untuk mendukung proses pengembangan game multiplayer, implementasi arsitektur

jaringan *client-server* dan *peer-to-peer*, serta pelaksanaan pengujian performa secara terkontrol pada jaringan lokal. Spesifikasi perangkat keras dirancang agar mampu menjalankan game secara stabil, memfasilitasi komunikasi jaringan *real-time*, serta memungkinkan pengukuran performa tanpa adanya *bottleneck* yang berasal dari keterbatasan perangkat, dalam penelitian ini spesifikasi perangkat keras disajikan pada tabel 3.1 dan 3.2.

Tabel 3.1 Spesifikasi Komputer

No.	Komponen	Spesifikasi
1.	Processor	Intel Core i5-4460
2.	RAM	8 GB
3.	Graphics	GT 1030 2GB
4.	Storage	512gb
5.	Network Adapter	MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
6.	Sistem Operasi	Windows 10

Perangkat utama yang dibutuhkan adalah 5 unit komputer yang berfungsi sebagai *node* dalam jaringan dan 1 unit komputer untuk berperan sebagai server dalam arsitektur client-server. Spesifikasi perangkat minimal mencakup prosesor *multi-core* untuk menangani proses logika game dan jaringan secara bersamaan, memori utama yang memadai untuk menjalankan Unity Editor maupun build game, serta kartu grafis yang mampu menangani rendering game secara stabil. Penggunaan perangkat dengan spesifikasi yang relatif homogen bertujuan untuk mengurangi variabel eksternal yang dapat memengaruhi hasil pengujian performa.

Tabel 3.2 Spesifikasi Router

No.	Komponen	Spesifikasi
1.	Model	ZTE ZXHN F609
2.	Tipe Perangkat	GPON ONT
3.	Standar Wi-Fi	IEEE 802.11b (Frekuensi 2.4 GHz)
4.	Kecepatan Wi-Fi	100 Mbps
5.	Antena	2 Antena Eksternal (Gain 5 dBi)
6.	Protokol Jaringan	IPv4
7.	Keamanan	WPA/WPA2-PSK
8.	Power Supply	Input: 100-240V AC; Output: 12V DC, 1A - 1.5A

Selain perangkat *client* dan server, dibutuhkan perangkat jaringan lokal berupa router yang berfungsi sebagai penghubung antar *node* dalam satu *Local Area Network*. Perangkat jaringan ini harus mendukung komunikasi dengan latensi rendah dan bandwidth yang stabil agar kondisi jaringan dapat dikontrol dan konsisten selama eksperimen. Seluruh perangkat dihubungkan melalui jaringan lokal yang sama dengan koneksi nirkabel dengan konfigurasi yang dijaga tetap konsisten pada seluruh skenario pengujian.

Perangkat keras harus mampu menjalankan aplikasi monitoring dan logging performa untuk mendukung pengujian performa, baik untuk pengukuran parameter jaringan maupun penggunaan sumber daya sistem. Oleh karena itu, spesifikasi perangkat perlu memastikan ketersediaan ruang penyimpanan yang cukup untuk menyimpan data log hasil pengujian serta stabilitas sistem selama pengujian dilakukan berulang kali. Dengan pemenuhan kebutuhan perangkat keras ini,

diharapkan hasil pengujian performa arsitektur *client-server* dan *peer-to-peer* dapat diperoleh secara objektif dan dapat dipertanggungjawabkan secara ilmiah.

3.2.4 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak dalam penelitian ini ditetapkan untuk mendukung proses pengembangan game *multiplayer*, implementasi dua arsitektur jaringan yang berbeda, serta pengujian dan evaluasi performa secara kuantitatif pada lingkungan jaringan lokal. Perangkat lunak yang digunakan dipilih berdasarkan stabilitas, relevansi dengan penelitian, serta kemampuannya dalam menyediakan data performa yang dapat dianalisis secara objektif, kebutuhan perangkat lunak yang digunakan dalam penelitian ini mencakup daftar perangkat lunak beserta fungsi dan perannya dalam penelitian disajikan pada tabel 3.3.

Tabel 3.3 Kebutuhan Perangkat Lunak

No.	Perangkat Lunak	Versi / Spesifikasi	Fungsi Utama	Peran dalam Penelitian
1.	Unity Engine	Unity LTS 6.3	Game engine untuk pengembangan game multiplayer	Pengembangan game dan integrasi networking
2.	C#	.NET Framework bawaan Unity	Bahasa pemrograman utama	Implementasi logika game dan sistem jaringan
3.	Mirror Networking	96.0.1	Framework networking berbasis client-server	Implementasi arsitektur client-server
4.	TCP/IP Native (.NET)	.NET Framework Bawaan Unity	Library networking berbasis TCP	Implementasi arsitektur peer-to-peer
5.	Python	3.13.5	Analisis performa runtime game	Pengukuran CPU usage dan memory usage

No.	Perangkat Lunak	Versi / Spesifikasi	Fungsi Utama	Peran dalam Penelitian
7.	Wireshark	4.6.4	Network protocol analyzer	Pengukuran latency, throughput, dan packet loss
8.	Sistem Operasi	Windows 11	Sistem operasi pengembangan dan pengujian	Platform eksekusi game dan tools

Game dikembangkan menggunakan Unity Engine sebagai game engine utama karena Unity menyediakan lingkungan pengembangan yang komprehensif, mendukung pengembangan game multiplayer, serta memiliki ekosistem *framework networking* yang luas. Unity digunakan untuk mengimplementasikan dua versi game dengan *gameplay* dan konten yang sama, sehingga perbedaan performa yang diamati dapat dikaitkan langsung dengan perbedaan arsitektur jaringan, bukan dengan perbedaan logika atau mekanik permainan. Bahasa pemrograman C# digunakan sebagai bahasa utama dalam pengembangan game dan integrasi sistem *networking*.

Implementasi arsitektur *client-server* pada penelitian ini menggunakan *Mirror Networking*. *Mirror* dipilih karena merupakan *framework open-source* yang masih aktif dikembangkan, bersifat ringan, serta mendukung arsitektur *client-server* dengan model *authoritative server*. *Mirror* menyediakan mekanisme sinkronisasi *state*, *Remote Procedure Call* (RPC), dan manajemen koneksi yang memungkinkan pengujian performa jaringan secara terkontrol pada jaringan lokal tanpa ketergantungan pada layanan *cloud* pihak ketiga.

Implementasi arsitektur peer-to-peer pada penelitian ini menggunakan TCP/IP *native* dari NET Framework (System.Net.Sockets) dengan pendekatan *custom implementation*, tanpa bergantung pada library networking pihak ketiga. Implementasi ini memanfaatkan TcpListener dan TcpClient untuk membangun arsitektur full mesh P2P, di mana setiap peer menjalankan TCP server sendiri dan terhubung langsung ke semua peer lain tanpa memerlukan server terpusat. Pendekatan ini memberikan kontrol penuh terhadap logika jaringan, meskipun TCP memiliki overhead yang lebih tinggi dibandingkan UDP, protokol ini menjamin *reliable delivery* dan *ordered packet transmission* yang penting untuk konsistensi data game, terutama untuk mekanisme kuis di mana setiap jawaban dan sinkronisasi state harus diterima dengan akurat oleh semua peer.

Dalam proses pengujian performa sistem, pengukuran penggunaan sumber daya komputasi dilakukan menggunakan script Python yang memanfaatkan modul psutil. Modul ini digunakan untuk memantau dan merekam penggunaan CPU serta penggunaan memori sistem selama game dijalankan pada setiap sesi pengujian. Script dijalankan secara paralel dengan aplikasi game dan mencatat data penggunaan sumber daya secara periodik sepanjang durasi eksperimen. Data yang dihasilkan kemudian disimpan dalam bentuk file csv sehingga dapat digunakan untuk analisis kuantitatif lebih lanjut. Untuk pengukuran performa jaringan, digunakan Wireshark sebagai network protocol analyzer untuk merekam lalu lintas paket data selama sesi permainan multiplayer berlangsung. Melalui proses packet capture, Wireshark memungkinkan analisis komunikasi jaringan pada level paket sehingga dapat diperoleh beberapa metrik penting, seperti latensi komunikasi,

throughput jaringan, dan packet loss pada masing-masing arsitektur jaringan yang diuji.

Sistem operasi yang digunakan adalah Microsoft Windows sebagai platform pengembangan dan pengujian, karena kompatibel dengan Unity, Mirror, NET Framework, serta seluruh perangkat lunak analisis performa yang digunakan. Seluruh perangkat lunak dikonfigurasikan pada lingkungan yang sama untuk memastikan konsistensi kondisi pengujian. Dengan pemenuhan kebutuhan perangkat lunak ini, penelitian dapat menghasilkan data performa yang objektif dan dapat digunakan untuk membandingkan arsitektur *client-server* dan *peer-to-peer* secara sistematis.

3.3 Lingkungan Pengujian dan Skenario Eksperimen

Tahap evaluasi merupakan tahap akhir dalam penelitian ini dan berfokus pada pengukuran serta perbandingan performa arsitektur *client-server* dan *peer-to-peer* yang telah diterapkan dalam game. Pengujian dilakukan pada lingkungan jaringan lokal nirkabel menggunakan perangkat yang telah ditentukan. Parameter performa yang diukur meliputi latensi komunikasi data, *throughput* jaringan, *packet loss*, jitter serta penggunaan sumber daya sistem berupa CPU, GPU, dan memori, secara keseluruhan parameter dan alat ukur disajikan pada tabel 3.4.

Tabel 3.4 Parameter dan Alat Ukur

No.	Parameter	Satuan	Alat Ukur
1.	Latensi	ms	Wireshark
2.	*Throughput*	KB/s	Wireshark
3.	*Jitter*	ms	Wireshark

4.	*Packet Loss*	%	Wireshark
5.	*CPU Usage*	%	Python
6.	*Memory Usage*	MB	Python
7.	*GPU Usage*	%	Python

Variabel independen dalam penelitian ini adalah jenis arsitektur jaringan yang digunakan yaitu *client-server* dan *peer-to-peer*. Variabel dependen meliputi metrik performa yang diukur yaitu latensi komunikasi, *throughput* jaringan, tingkat *packet loss*, penggunaan CPU, dan penggunaan memori. Untuk menjaga validitas hasil penelitian, sejumlah variabel kontrol diterapkan, seperti spesifikasi perangkat keras, konfigurasi jaringan, konten game, serta durasi dan skenario pengujian yang dibuat identik pada kedua arsitektur, secara keseluruhan variabel penlitian disajikan pada tabel 3.5.

Tabel 3.5 Variabel Penelitian

No.	**Jenis Variabel**	**Keterangan**
1.	Variabel Bebas	Arsitektur jaringan
2.	Variabel Terikat	Latensi, *throughput*, *packet loss*, CPU, GPU, memori
3.	Variabel Kontrol	Perangkat, game, durasi pengujian, jumlah pemain

Pengujian dilakukan dalam lingkungan jaringan lokal nirkabel untuk merepresentasikan kondisi penggunaan game *multiplayer* LAN secara realistis degan kondisi skenario 2 pemain dan 5 pemain. Setiap arsitektur jaringan diuji pada kondisi jaringan yang sama untuk menjaga konsistensi pengukuran seperti yang ditunjukkan pada tabel 7

Tabel 3.6 Skenario Eksperimen

No.	Skenario	Client-Server	P2P	Durasi	Repetisi
1.	*S1-Light*	2 Client	2 Peer	5 Menit	3
2.	*S2-Full*	5 Client	5 Peer	5 Menit	3

Skenario eksperimen dirancang dengan jumlah pemain 2 player dan 5 player dengan aktivitas permainan yang identik pada setiap sesi pengujian. Setiap skenario dijalankan dalam 3 kali pengulangan untuk meningkatkan reliabilitas data karena pada sistem jaringan nirkabel performa sangat dipengaruhi oleh faktor-faktor dinamis seperti fluktuasi sinyal, interferensi kanal, dan proses *scheduling* sistem operasi sehingga hasil pengujian tunggal berpotensi bias dan tidak representatif (Kotz dkk., 2004). Jumlah tiga kali pengulangan dipilih sebagai standar minimum untuk memperoleh nilai rata-rata yang stabil dan konsisten sekaligus menjaga efisiensi durasi penelitian tanpa mengurangi validitas hasil pengujian secara keseluruhan (Uta dkk., 2020). Selama pengujian berlangsung data performa sistem dan jaringan direkam secara simultan. Data penelitian dikumpulkan melalui pengukuran langsung selama sesi permainan berlangsung. Data performa sistem dicatat menggunakan *python* dengan modul psutil, sementara data jaringan direkam menggunakan *Wireshark*. Seluruh data direkam secara otomatis dan disimpan dalam format CSV untuk memudahkan proses analisis lanjutan. Pendekatan ini memastikan bahwa data yang diperoleh bersifat objektif dan dapat diverifikasi.

Data yang terkumpul dianalisis menggunakan statistik deskriptif untuk memperoleh gambaran umum performa masing-masing arsitektur, seperti nilai rata-rata, standar deviasi, dan rentang nilai. Selanjutnya, analisis komparatif dilakukan

untuk menguji perbedaan performa antara arsitektur *client-server* dan *peer-to-peer* menggunakan uji statistik yang sesuai. Hasil analisis divisualisasikan dalam bentuk grafik dan tabel untuk mempermudah interpretasi.

3.4 Evaluasi Performa

Evaluasi performa pada penelitian ini dilakukan menggunakan pendekatan kuantitatif dengan menghitung nilai statistik dasar dari data hasil pengujian. Setiap parameter performa dihitung berdasarkan data mentah yang diperoleh dari tiga kali replikasi pengujian pada masing-masing skenario jaringan dengan *Wi-Fi* normal. Pendekatan ini bertujuan untuk memperoleh nilai performa yang representatif dan mengurangi pengaruh anomali atau fluktuasi jaringan sesaat.

Setiap pengujian dilakukan sebanyak tiga kali dengan kondisi eksperimen yang sama. Nilai akhir setiap parameter performa diperoleh dari nilai rata-rata ketiga replikasi pengujian tersebut. Secara matematis, nilai rata-rata untuk suatu parameter performa dinyatakan sebagai:

$$\bar{X} = Average\ (X_1 + X_2 + \cdots + X_n)$$

dengan:

$\bar{X}$ = Nilai rata-rata parameter performa

$X_1 + X_2 + X_3$ = Hasil pengujian ke-1, ke-2, ke-3

Nilai rata-rata ini digunakan sebagai nilai representatif untuk masing-masing skenario jaringan dan arsitektur.

Perhitungan *latency* yang merepresentasikan waktu tunda komunikasi data antara pengirim dan penerima dianalisis berdasarkan selisih waktu pengiriman dan penerimaan paket data jaringan yang direkam menggunakan *Wireshark*. *Latency* satu paket dihitung menggunakan persamaan:

$$L_i = T_{receive,i} - T_{send,i}$$

dengan:

L_i = latency paket ke-i (ms)

$T_{receive,i}$ = waktu paket diterima

$T_{send,i}$ = waktu paket dikirim

Nilai latency rata-rata selama satu sesi pengujian dihitung sebagai:

$$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i$$

dengan:

$\bar{L}$= latency rata-rata

n= jumlah paket yang dianalisis

Latency rata-rata ini kemudian dirata-ratakan kembali dari tiga replikasi pengujian untuk memperoleh nilai akhir *latency* pada masing-masing skenario.

Jitter merupakan variasi atau fluktuasi delay antar paket yang diterima selama proses komunikasi jaringan berlangsung. Pada aplikasi real-time seperti game multiplayer, nilai jitter yang tinggi dapat menyebabkan ketidakkonsistenan

sinkronisasi data antar pemain sehingga mempengaruhi kelancaran interaksi permainan. Secara matematis, jitter dihitung berdasarkan selisih delay antar paket yang berurutan. Jika delay paket ke-i dinyatakan sebagai D_i, maka jitter antar dua paket berturut-turut dapat dihitung menggunakan persamaan:

$$J_i = | D_i - D_{i-1} |$$

dengan:

J_i= jitter antara paket ke-i dan paket sebelumnya

D_i= delay paket ke-i

D_{i-1}= delay paket ke-(i-1)

Untuk memperoleh nilai jitter rata-rata selama satu sesi pengujian, digunakan persamaan:

$$J_{avg} = \frac{1}{n-1} \sum_{i=2}^{n} | D_i - D_{i-1} |$$

dengan:

J_{avg}= rata-rata jitter

n= jumlah paket yang dianalisis

D_i= delay paket ke-i

Nilai jitter yang lebih kecil menunjukkan bahwa variasi delay antar paket relatif stabil, sedangkan nilai jitter yang besar menunjukkan fluktuasi delay yang tinggi pada komunikasi jaringan.

Throughput menunjukkan jumlah data yang berhasil dikirimkan melalui jaringan per satuan waktu. Nilai *throughput* dihitung berdasarkan total ukuran data yang diterima selama sesi permainan. Secara matematis, *throughput* dihitung menggunakan rumus:

$$T = \frac{D_{total}}{t_{total}}$$

dengan:

T= throughput (bps atau Kbps)

D_{total}= total data yang diterima (bit)

t_{total}= durasi pengujian (detik)

Jika data diperoleh dalam satuan byte, maka dilakukan konversi ke bit, nilai throughput akhir diperoleh dari rata-rata hasil tiga kali pengujian.

Packet loss menggambarkan persentase paket data yang hilang selama transmisi. Parameter ini penting untuk menilai keandalan komunikasi jaringan. *Packet loss* dihitung menggunakan persamaan:

$$PL = \left(\frac{P_{send} - P_{receive}}{P_{send}}\right) \times 100\%$$

dengan:

PL= packet loss (%)

P_{send}= jumlah paket yang dikirim

$P_{receive}$= jumlah paket yang diterima

Nilai *packet loss* dihitung untuk setiap sesi pengujian kemudian dirata-ratakan dari tiga replikasi pengujian.

Penggunaan CPU direkam secara periodik selama sesi pengujian, untuk memperoleh nilai penggunaan CPU yang representatif, digunakan nilai rata-rata waktu CPU dari seluruh data yang direkam selama sesi pengujian.

$$CPU_{avg} = \frac{1}{n}\sum_{i=1}^{n} CP\ U_i$$

dengan:

CPU_{avg}= rata-rata penggunaan CPU (%)

CPU_i= penggunaan CPU pada sample ke-i (%)

n= jumlah sample pengukuran selama sesi pengujian

Nilai ini digunakan sebagai indikator beban penggunaan CPU pada saat pengujian dilaksanakan untuk kedua arsitektur jaringan.

GPU usage digunakan untuk mengukur tingkat pemanfaatan unit pemrosesan grafis selama game dijalankan. Parameter ini penting untuk mengetahui beban komputasi grafis yang dihasilkan oleh masing-masing arsitektur jaringan ketika game multiplayer berjalan secara real-time. Penggunaan GPU direkam secara periodik selama sesi pengujian dengan interval tertentu, maka nilai penggunaan GPU rata-rata dapat dihitung menggunakan persamaan:

$$GPU_{avg} = \frac{1}{n}\sum_{i=1}^{n} GPU_i$$

dengan:

GPU_{avg}= rata-rata penggunaan GPU (%)

GPU_i= penggunaan GPU pada sampel ke-i (%)

n= jumlah sampel pengukuran selama sesi pengujian

Nilai rata-rata GPU menunjukkan tingkat beban grafis secara umum selama permainan.

Penggunaan memori dianalisis berdasarkan data total *allocated memory* yang direkam selama sesi permainan. Nilai memori yang digunakan tidak diambil dari satu titik waktu tertentu, melainkan dihitung sebagai nilai rata-rata selama durasi pengujian. Secara matematis, rata-rata penggunaan memori dihitung sebagai:

$$Mem_{avg} = \frac{1}{t}\sum_{i=1}^{t} Mem_i$$

dengan:

Mem_{avg}= rata-rata penggunaan memori (MB)

Mem_i= penggunaan memori pada waktu ke-i

t= jumlah sampel waktu

Pendekatan ini memastikan bahwa nilai memori mencerminkan kebutuhan sumber daya aplikasi secara keseluruhan selama sesi permainan.

Dalam proses membandingkan performa antar arsitektur, selisih performa dihitung menggunakan persamaan perbedaan relatif:

$$\Delta X = X_{CS} - X_{P2P}$$

dengan:

ΔX= selisih parameter performa

X_{CS}= nilai performa arsitektur client-server

X_{P2P}= nilai performa arsitektur peer-to-peer

Pemilihan arsitektur *client-server* sebagai nilai referensi didasarkan pada pertimbangan metodologis dan konvensi ilmiah dalam penelitian sistem jaringan. Arsitektur *client-server* merupakan model yang paling umum, matang, dan luas digunakan dalam implementasi game *multiplayer* modern, baik pada jaringan lokal maupun internet. Oleh karena itu, *client-server* diperlakukan sebagai arsitektur pembanding standar terhadap pendekatan alternatif, yaitu *peer-to-peer*. Dengan menetapkan *client-server* sebagai *baseline*, nilai ΔX memiliki interpretasi yang jelas dan konsisten, jika $\Delta X > 0$, maka performa *client-server* lebih tinggi dibandingkan *peer-to-peer* untuk parameter tertentu, Jika $\Delta X < 0$, maka *peer-to-peer* menunjukkan performa yang lebih baik dibandingkan *client-server*, Jika $\Delta X = 0$, maka kedua arsitektur memiliki performa yang setara. Dengan metode evaluasi ini, penelitian diharapkan dapat memberikan gambaran yang objektif dan terukur mengenai kelebihan dan keterbatasan masing-masing arsitektur jaringan dalam mendukung performa game *multiplayer* lokal, khususnya pada kondisi jaringan nirkabel.