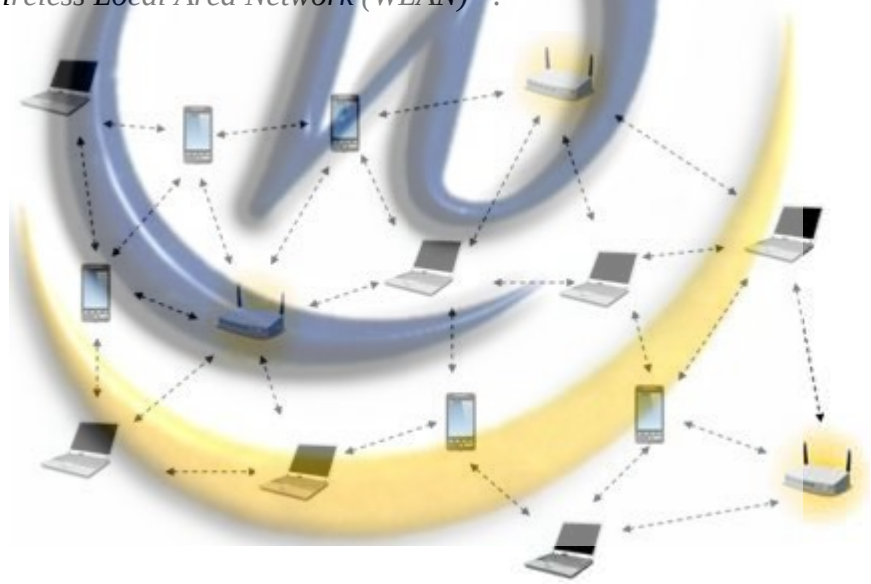
BAB II

LANDASAN TEORI

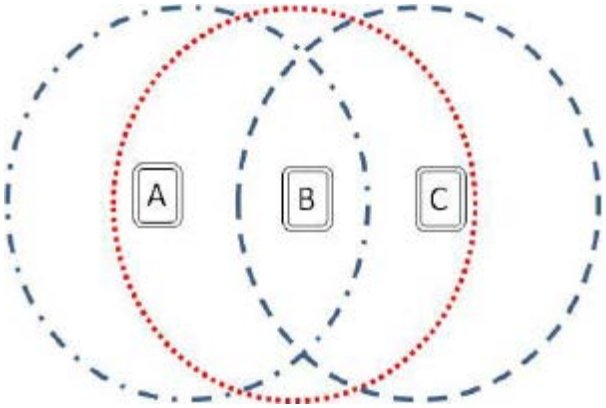
2.1 *Mobile Ad-hoc Network (MANET)*

Mobile Ad-hoc Network (MANET) adalah kumpulan *node* bergerak nirkabel yang berkomunikasi satu sama lain menggunakan saluran nirkabel *multihop* tanpa infrastruktur jaringan yang ada atau administrasi terpusat. Setiap perangkat memiliki antarmuka nirkabel dan saling berkomunikasi melalui gelombang radio, setiap perangkat tersebut dinamakan node. Jaringan *MANET* tidak lagi memerlukan *router-router* yang tetap (*fixed-router)* maupun lokasi yang tetap (*fixed-location)* pada infrastukturnya, seperti terlihat pada gambar dibawah. Contoh infrastruktur jaringan adalah jaringan selular, *Local Area Network (LAN)* atau *Wireless Local Area Network (WLAN)*[2].



Gambar 2.1 Struktur Jaringan *MANET*

Pesan yang melewati node di *MANET* dilakukan dengan menggunakan jalur *multihop*. Setiap node di *MANET* adalah *host* dan *router* yang berbagi media nirkabel dan bebas bergerak ke segala arah[3].



Gambar 2.2 *MANET* dengan Tiga Node

Gambar 2.2 menunjukkan contoh sederhana dari *MANET* dengan tiga buah node yang terdiri dari node A, node B dan node C. Asumsikan node A dan node C tidak berada dalam jangkauan yang sama untuk bertukar informasi, namun node B dapat digunakan untuk meneruskan paket data antara node A dan node C karena node B berada dalam jangkauan node A dan node C dimana node B akan bertindak sebagar *router* dari ketiga node ini untuk membentuk jaringan *ad-hoc* yang memiliki jalur bernama A-B-C.

Dalam jalur *multi-hop*, node *slave* mengirimkan paket data menuju node *master* melalui node-node terdekat di sekitarnya. Dalam jaringan node *mobile* yang ada pada *MANET* akan rentan terputus dikarenakan oleh mobilitas dari setiap node tersebut sehingga kecepatan node berbanding lurus dengan jumlah rute yang rusak. Jalur *multi-hop* harus bebas dari kegagalan rute untuk transmisi paket data dari sumber mana pun ke tujuan mana pun. Setiap rute yang rusak dalam suatu jaringan harus ditangani dengan cepat oleh rute baru sehingga membuat paket data mencapai tempat tujuan dengan sukses. Oleh karena itu, protokol *routing* sangat diperlukan pada *MANET* untuk menangani kegegalan rute. Faktor yang mempengaruhi kinerja *MANET* adalah ukuran jaringan, model mobilitas, jenis protokol *routing*, kecepatan, dan dapat dijelaskan dalam bentuk *end to end delay*, *packet delivery ratio*, dan *throughput*[3].

MANET dapat digunakan untuk berbagai jenis area diantaranya[4] :

1. Lingkungan militer
    - Medan perang otomatis
    - Operasi khusus
    - Pertahanan dalam negeri

2. Lingkungan sipil
   - Pemulihan bencana (banjir, kebakaran, gempa bumi)
   - Penegak hukum (kontrol kerumunan)
   - Pemantauan untuk pencarian dan pertolongan di daerah-daerah terpencil (sensor)
   - Stadion olahraga
3. Komersial
   - Acara olahraga, festival, konvensi
   - Pemantauan pasien
   - Komputasi kolaboratif ad hoc (*Bluetooth*)
   - Sensor pada mobil (keselamatan navigasi mobil)
   - Komunikasi antar kendaraan

2.1.1 Karakteristik *MANET*

Karakteristik *MANET* ditandai oleh beberapa fitur spesifik, diantaranya[4]:

1. Nirkabel

   Node–node terhubung melalui jaringan nirkabel dan komunikasi yang dilakukan di antara node–node tersebut juga menggunakan jaringan nirkabel.
2. Self-organizing dan self-managing

   Node-node dapat mengelola dan memelihara sendiri.
3. Berbasis *ad-hoc*

   MANET adalah jaringan yang dibentuk sesuai kebuhan node yang bersifat sementara dan dinamis.
4. Topologi dinamis

   Karena node–node bergerak ke segala arah dengan kecepatan yang bervariasi, maka topologi jaringan akan berubah secara acak dan tak terduga.
5. *Routing multi-hop*

   Tidak ada *router* khusus dan setiap node bertindak sebagai router untuk mengirimkan paket ke tujuan node berikutnya.

6. Tidak bergantung pada infrastruktur

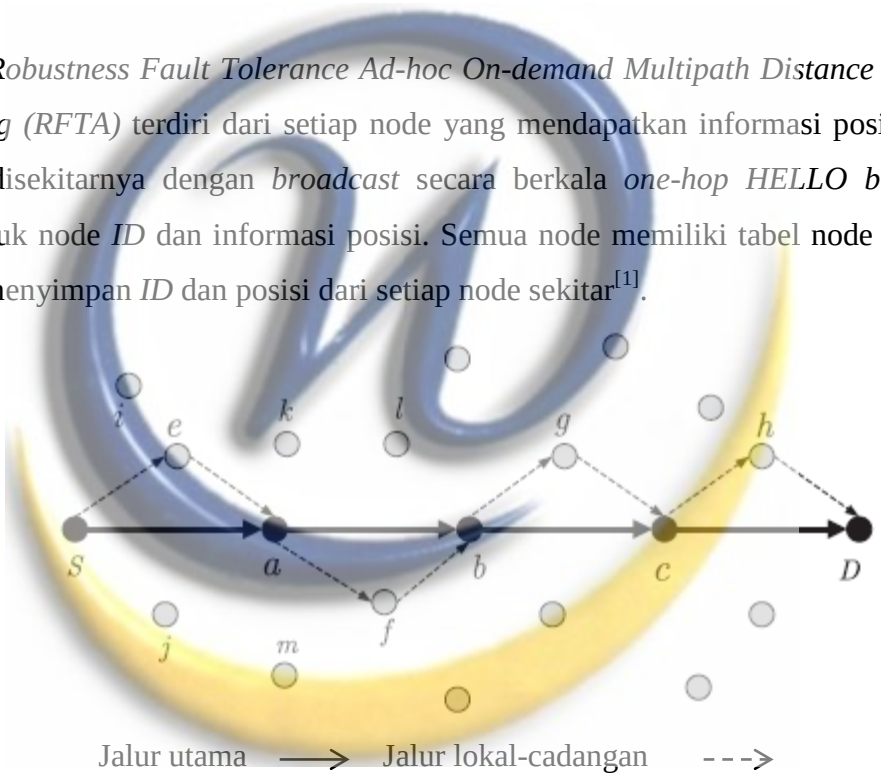
   Jaringan akan mengatur secara mandiri dan tidak bergantung pada apa infrastruktur atau kontrol terpusat. Mode operasi dari setiap node mampu bertindak sebagai router independen dan menghasilkan data independen.

7. Keterbatasan energi

   Keterbatasani energi menjadi masalah utama karena disaat node berpindah, node tidak mendapat konsumsi daya listrik sehingga menggunakan baterai yang memiliki keterbatasan.

2.2 *RFTA*

Robustness Fault Tolerance Ad-hoc On-demand Multipath Distance Vector Routing (RFTA) terdiri dari setiap node yang mendapatkan informasi posisi dari node disekitarnya dengan *broadcast* secara berkala *one-hop HELLO beacons* termasuk node *ID* dan informasi posisi. Semua node memiliki tabel node sekitar yang menyimpan *ID* dan posisi dari setiap node sekitar[1].



Gambar 2.3 Penentuan Rute *RFTA*

Setiap node memiliki alamat *IP* atau *ID* unik. Pada gambar 2.3 terdapat S yang melambangkan sumber dan D yang melambangkan tujuan. Penentuan rute sudah dimulai dari sebelum S mengirimkan paket data ke D walaupun terjadi kendala pada jalur rute ke D. Penerusan pengiriman paket data dilihat dari posisi node terdekat dan tujuan paket untuk membuat keputusan paket *forwarding*. Ketika terjadi kendala pada jalur utama, node sebelumnya akan mengetahui posisi node tedeka pada jalur lokal-cadangan, pilihan jalur cadangan yang optimal dari hop berikutnya adalah node yang secara geografis paling dekat dengan tujuan

paket. Dalam hal ini, setiap paket *RREQ* memiliki struktur yang sama, termasuk *ID* dan posisi tujuan, sehingga nilai awal dari masa pakai tautan di *RREQ* ditetapkan sebagai 0[1].

2.2.1 Pseudo Code *RFTA*

Langkah yang digunakan oleh *source node*, *destination node* dan *intermediate node* dijelaskan di bawah ini :

Langkah Algoritma *Source Node* pada *RFTA*[5] :

1. Jika ada data yang akan dikirim menuju destinasi tertentu dan tidak ada *valid path* untuk destinasi tersebut, *broadcast RREQ packet.*
2. Tunggu untuk *RREP packet* yang pertama datang.
3. Setelah menerima *RREP packet* yang pertama, tunggu untuk beberapa saat, kemudian dari *path* yang diterima pilih salah satu yang memiliki *active neighbors* paling sedikit dan memulai *data load balancing* pada *path* tersebut.

Langkah Algoritma *Destination Node* pada *RFTA*[5] :

1. Kirim kembali *RREP packet* ke seluruh *node* dari mana *RREQ packet* diterima.

Langkah Algoritma *Intermediate Node* pada *RFTA*[5] :

1. Setelah *RREQ message* diterima dan *acceptable* lakukan :
   i. Simpan *message* pada tabel *RREQ_Seen.*
   ii. Buat *RREQ_Query packet.*
   iii. Kirim *RREQ_Query packet* ke *neighbours* dan mengkonfirmasi apakah sudah melihat *RREQ message* yang sama sebelumnya.
   iv. Tunggu beberapa saat untuk *RREQ_Query_Reply messages* dari *neighbours.*
   v. Tambahkan *ActiveNeighborCount* berdasarkan respon positif yang diterima.
   vi. *Broadcast RREQ message* dengan *ActiveNeighborCount value* yang baru.
2. Setelah *RREQ_Query message* diterima, salah satu langkah di bawah akan dilakukan :
   i. Apabila pada *RREQ_Seen table* ini merupakan *RREQ message* baru. Simpan *RREQ* di *RREQ_Seen table.*

ii. Apabila pada *RREQ_Seen table* ini merupakan *RREQ* yang diterima dari *node* yang sama, abaikan *query*.

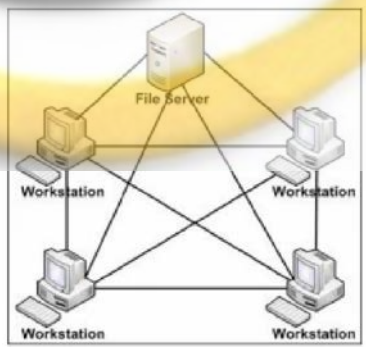
iii. Apabila *RREQ* merupakan *RREQ* yang sama dengan yang ada di *RREQ_Seen table* dan diterima dari *neighbor* yang baru tetapi *RREQ* yang sama belum di *broadcast* sebelumnya, maka hanya merespon dengan mengirim kembali *positive RREQ_Query_Reply*.

iv. Apabila *RREQ* merupakan *RREQ* yang sama dengan yang ada di *RREQ_Seen table* dan diterima dari *neighbor* yang baru dan *RREQ* yang sama telah di *broadcast* sebelumnya, maka respon dengan mengirim kembali *positive RREQ_Query_Replypositive RREQ_Query_Reply* dan tambahkan *RREQ* yang sebelumnya ke *A_N_C field* pada *RREQ_Seen table.*

Setelah *RREP message* diterima, tambahkan *RREP* yang sebelumnya ke *N_C* pada *field ActiveNeighborCount* pada *RREP table* dan kirim kembali *RREP*

2.3 Topologi *MESH*

Topologi *MESH* merupakan salah satu jenis topologi jaringan yang terdiri dari node–node membentuk rangkaian seperti jaring/jala dimana setiap node saling terhubung satu sama lain di dalam jaringan sehingga ketika berkomunikasi setiap node tidak memerlukan perantara[6].



Gambar 2.4 Contoh Topologi *MESH*

Topologi *MESH* memiliki beberapa karakteristik atau ciri-ciri diantaranya adalah:

1 Setiap perangkat yang ada di dalam jaringan akan saling terhubung satu sama lain.

2 Kabel yang digunakan dalam berkomunikasi secara langsung dengan node lainnya di dalam jaringan cukup banyak.

3 Pada setiap node memiliki setidaknya 2 atau lebih dari port I/O.

4 Setiap node memiliki konfigurasi yang berbeda dalam berkomunikasi.

2.4 Parameter *Quality of Service (QoS)*

Quality of Service (QoS) adalah suatu metode untuk menentukan seberapa baik kinerja jaringan yang dilihat dari hasil pengukuran beberapa parameter performansi yang digunakan sebagai acuan sehingga *QoS* juga dapat mendefinisikan karakteristik dan sifat dari suatu layanan yang sedang digunakan[7].

Kinerja jaringan dapat menunjukan konsistensi, tingkat keberhasilan pengiriman data, dan lain-lain. Ada beberapa parameter yang dapat digunakan untuk mengukur kinerja jaringan antara lain[8] :

1. *Packet delivery ratio*
2. *Jitter*
3. *Throughput*
4. *Delay*
5. *Packet Loss*
6. *Routing overhead*

Parameter – parameter yang akan digunakan dalam penelitian ini adalah:

2.4.1 End to end delay

Delay atau dikenal dengan istilah *latency* adalah waktu rata–rata yang dibutuhkan oleh sebuah data untuk mencapai tujuan. Dalam proses pengiriman data, terdapat beberapa faktor yang mempengaruhi pengiriman data sehingga mengakibatkan *delay*. Faktor tersebut antara lain jarak antar node, waktu proses data, proses pencarian rute dan media yang dilalui oleh data[7]. Pada Tabel 2.1 diperlihatkan kategori dari *delay* yang diambil dari standar TIPHON (*Telecommunications and Internet Protocol Harmonization Over Networks*)[9].

Tabel 2.1 Kategori *Delay*

Kategori *Delay*	Besar *Delay*(ms)	Indeks
Sangat Baik	< 150 ms	4
Baik	150 ms s/d 300 ms	3
Sedang	300 ms s/d 450 ms	2
Jelek	> 450 ms	1

Adapun *average end to end delay* dapat dihitung dengan rumus berikut:

$$Average_{Delay} = \frac{\sum \text{waktu}_{\text{diterima}} - \text{waktu}_{\text{dikirim}}}{\sum \text{banyaknya}_{\text{koneksi}}} \dots (7)$$

2.4.2 Packet delivery ratio

Packet delivery ratio adalah rasio antara total paket yang diterima dengan total paket yang dikirim. *Packet delivery ratio* juga dapat menunjukkan tingkat keberhasilan protokol *routing* dimana jika nilai *packet delivery ratio* tinggi, maka tingkat keberhasilan protokol *routing* akan tinggi dalam melakukan mengirimkan paket begitu pula sebaliknya. Adapun *packet delivery ratio* dapat dihitung dengan rumus berikut[7] :

$$Packet\_Delivery\_Ratio = \frac{\sum \text{paket\_diterima}}{\sum \text{paket\_dikirim}} \text{x } 100\% \dots (7)$$

2.4.3 Throughput

Throughput adalah kecepatan (*rate*) pada transfer data yang berhasil melalui sebuah kanal komunikasi dalam selang waktu tertentu dengan satuan *byte per second (bps)*. Semakin tinggi nilai *throughput*, maka jaringan akan memiliki performa yang lebih baik. Adapun *throughput* dapat dihitung dengan rumus berikut[7] :

$$Throughput = \frac{\sum \text{paket\_data\_yang\_diterima}}{\sum \text{waktu\_simulasi}} \dots (7)$$

2.4.4 Routing overhead

Routing overhead adalah rasio antara paket *routing* yang di transmisikan oleh protokol *routing* terhadap paket data yang berhasil dikirim. *Routing overhead* dapat digunakan untuk menghitung efisiensi kerja suatu protokol *routing*. Adapaun *routing overhead* dapat dihitung dengan rumus berikut[7] :

$$Routing\_Overhead = \frac{\sum \text{paket\_}routing\text{\_yang\_diterima}}{\sum \text{paket\_yang\_dikirim}} \dots (7)$$

2.5 Metode *Prototyping*

Metode *prototyping* merupakan metode pengembangan yang cepat dan pengujian terhadap model kerja (*prototipe*) dari aplikasi baru melalui proses interaksi secara berulang-ulang yang biasa digunakan ahli sistem informasi dan ahli bisnis (O'Brien, 2005). Proses pembuatan *prototyping* merupakan proses yang interaktif dan berulang-ulang yang menggabungkan langkah-langkah siklus pengembangan tradisional. *Prototyping* dievaluasi beberapa kali sebelum pemakai akhir menyatakan *prototyping* tersebut diterima. Adapun langkah-langkah *prototyping* adalah sebagai berikut[2] :

1. Analisis Kebutuhan Sistem

   Pembangunan sistem memerlukan penyelidikan dan analisis mengenai alasan timbulnya ide atau gagasan untuk membangun dan mengembangkan sistem dan mendeskripsikan apa yang harus dilakukan sistem untuk memenuhi kebutuhan informasi.

2. Desain Sistem

   Desain sistem menentukan bagaimana sistem akan memenuhi tujuan tersebut yang terdiri dari aktivitas desain yang menghasilkan spesifikasi fungsional seperti desain *interface* maupun topologi.

3. Pengujian Sistem

   Sistem yang telah dirancang akan diuji, diimplementasikan, dievaluasi dan dimodifikasi berulang-ulang hingga dapat diterima pemakainya (O'Brien, 2005). Pengujian sistem bertujuan menemukan kesalahan-kesalahan yang terjadi pada sistem dan melakukan revisi sistem[2].

2.6 Simulasi Jaringan

Simulasi jaringan adalah teknik untuk memeragakan perilaku jaringan dengan menghitung interaksi antara entitas jaringan (node, paket, data, router) yang berbeda dengan menggunakan metode tertentu[10]. Simulasi jaringan dibangun untuk mengambil data yang akan diolah pada analisis[12].

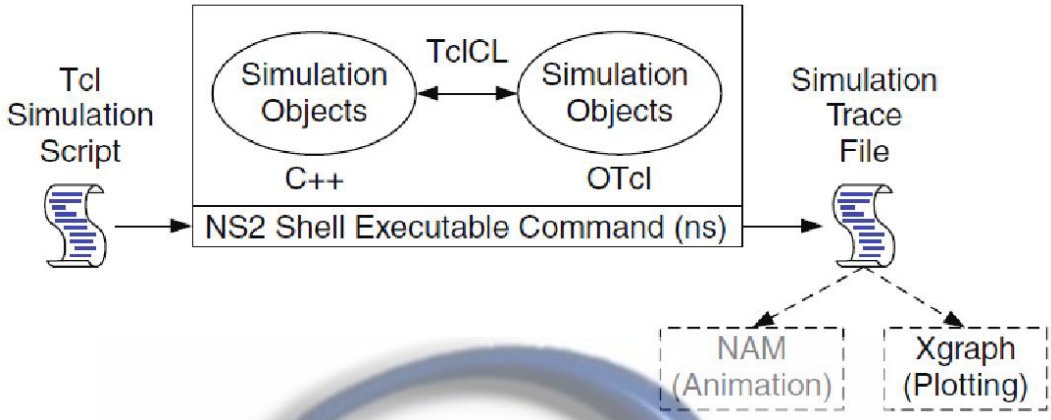
Keuntungan dengan melakukan simulasi jaringan adalah tidak menyebabkan permasalahan pada jaringan yang sesungguhnya. Oleh karena itu kondisi jaringan dan berbagai aplikasi dan layanan yang mendukung dapat diuji dan diamati secara leluasa di laboratorium melalui simulator, berbagai atribut lingkungan juga dapat dimodifikasi dengan cara yang terkontrol untuk menilai bagaimana jaringan akan berperilaku dibawah kondisi yang berbeda[10].

2.7 *Network Simulator*

Network Simulator atau NS merupakan salah satu perangkat lunak untuk membangun simulasi jaringan berbasis *TCP/IP* dengan berbagai macam medianya. Selain itu juga dapat digunakan untuk mensimulasikan protokol jaringan (*TCP/UDP/RTP*), *Traffic behavior* (*FTP*, *Telnet, CBR*, dan lain-lain), *Queue management (RED, FIFO, CBQ)*, algoritma *routing unicast (Distance Vector, Link State)* dan *multicast (PIM SM, PIM DM, DVRMP, Shared Tree* dan *Bidirectional Shared Tree)*, aplikasi multimedia yang berupa *layered video, Quality of Service, video-audio dan transcoding*. *Network Simulator* juga mengimplementasikan beberapa *MAC* (*IEEE* 802.3, 802.11) di berbagai media, misalnya jaringan berkabel (seperti *LAN, WAN, point to point*), jaringan tanpa kabel (seperti *mobile IP*, *Wireless LAN*), bahkan simulasi hubungan antar node jaringan yang menggunakan media satelit[11].

Adapun beberapa keuntungan dengan menggunakan *Network Simulator* sebagai perangkat lunak simulasi pembantu analisi dalam riset, antara lain adalah *Network Simulator* dilengkapi dengan *tool* validasi yang digunakan untuk menguji validitas pemodelan yang ada pada *Network Simulator*. Pembuatan simulasi dengan menggunakan *Network Simulator* jauh lebih mudah karena pemodelan media, protokol dan *network component* lengkap dengan perilaku trafiknya sudah tersedia pada *library Network Simulator*. *Network Simulator* juga bersifat *open source* dibawah *Gnu Public License (GPL)*, sehingga *Network Simulator* dapat di *download* dan digunakan secara gratis melalui *website* yaitu http://www.isi.edu/nsnam/.[11]

Network Simulator dibangun dengan menggunakan 2 bahasa pemograman yaitu *C++* dan *OTcl* (*Object-orientes Tool Command Language*). Arsitektur dasar dari *Network Simulator* dapat digambarkan seperti berikut[12] :



Gambar 2.5 Arsitektur Dasar *Network Simulator*

C++ digunakan untuk *library* yang mendefinisikan mekanisme internal dari objek simulasi, sedangkan *OTcl* digunakan pada *script* simulasi untuk menangani interaksi langsung antara pengguna dengan simulator serta menangani interaksi antara objek-objek *OTcl* lainnya. C++ dan *OTcl* saling terhubung dengan menggunakan komponen *TclCL*. Variabel-variabel pada domain *OTcl* dipetakan pada objek C++. Variabel ini cenderung dikenal sebagai sebuah *handle*. Dalam domain *OTcl*, sebuah *handle* berfungsi sebagai substansi untuk menangani interaksi simulator dengan pengguna. Hasil yang dikeluarkan oleh *Network Simulator* berupa file *trace* dan harus diproses dengan menggunakan *tool* lain, seperti *Network Animator* (NAM), Xgraph, *perl*, *awk*, maupun *gnuplot*[11].

2.8 *TCP\IP*

TCP/IP (*Transmission Control Protocol/Internet Protocol*) adalah gabungan dari protokol *TCP* (*Transmission Control Protocol*) dan *IP* (*Internet Protocol*) sebagai sekelompok protokol yang mengatur komunikasi data dalam proses tukar-menukar data dari satu komputer ke komputer lain di dalam jaringan internet yang akan memastikan pengiriman data sampai ke alamat yang dituju.

Protokol adalah sebuah aturan atau standar yang mengatur atau mengijinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol perlu diutamakan pada penggunaan standar teknis, untuk

menspesifikasi bagaimana membangun komputer atau menghubungkan peralatan perangkat keras. Protokol secara umum digunakan pada komunikasi real-time dimana standar digunakan untuk mengatur struktur dari informasi untuk penyimpanan jangka panjang.

2.9 *OSI Layer*

OSI singkatan dari *Open System Interconnection* adalah sebuah model arsitektural jaringan yang dikembangkan oleh badan *International Organization for Standardization (ISO)* di Eropa pada tahun 1977 hingga akhirnya *OSI* juga dikenal dengan *OSI seven layer model*. Berikut 7 model *OSI layer*.

1. *Application Layer*

Layer paling atas dari model *OSI* adalah lapisan aplikasi, kapasitas dari lapisan ini adalah menangani isu masalah seperti transparansi data, mengalokasikan sumber daya. Sebagai salah satu *layer* atau lapisan yang paling penting, karena menampilkan konten dan juga informasi di dalam jarignan kepada usernya.

Maka dari itu, *application layer* pun dibantu dengan beberapa protokol dalam menjalankan tugas dan juga fungsinya salah satunya adalah *FTP*. *FTP* merupakan kependekan dari *File Transfer Protocol* yang memiliki fungsi untuk melakukan proses pengiriman *file* dari satu komputer ke komputer yang lain, yang terhubung ke dalam satu jaringan.

2. *Presentation Layer*

Presentation layer merupakan sebuah lapisan yang berada pada model lapisan *OSI layer*, yang berada pada level atau lapisan kedua ketika sebuah data atau informasi akan dikirim, dan berada pada *layer* ke enam pada saat sebuah data akan diterima oleh *user*.

3. *Session Layer*

Session layer merupakan *layer* atau lapisan kelima dari keseluruhan lapisan *OSI layer* pada saat *user* menerima data dari sebuah jaringan dan bertindak sebagai layer ketiga ketika terjadi pengiriman sebuah koneksi atau paket data (peran sebagai *transmitter*).

4. *Transport Layer*

Transport layer ini dapat menggabungkan beberapa koneksi transport ke dalam jaringan koneksi yang sama. *Transport layer* bertanggung jawab untuk menyampaikan data ke proses aplikasi yang sesuai pada komputer user.

5. *Network Layer*

Network layer adalah lapisan yang menyediakan sarana fungsional dan prosedural untuk mentransfer beberapa variabel data secara berurutan (disebut datagrams) dari satu titik ke titik lainnya yang terhubung ke jaringan yang sama.

6. *Data Link Layer*

Data link layer adalah lapisan yang memiliki tugas utama untuk menyediakan sebuah prosedur pengiriman data antar jaringan. Dalam proses transmisi data yang terjadi, *data link layer* merupakan layer keenam untuk transmitter atau pengirim data, dan merupakan layer kedua untuk receiver, atau yang menerima data.

7. *Physical Layer*

Physical layer merupakan lapisan atau *layer* ketujuh ketika sebuah paket data mulai ditransmisikan oleh *transmitter*. Sebagai contoh dalam hal ini adalah sebuah server computer yang merupakan lapisan yang pertama kali harus dilewati oleh paket data atau informasi ketika akan melakkan proses penerimaan oleh *receiver*.