BAB II
LANDASAN TEORI

2.1. Komunitas Trail Adventure Balikpapan

Pulau Kalimantan selalu menyajikan alam yang menarik untuk dijelajahi, baik melalui darat, laut, maupun udara. Citra Kalimantan sebagai rimba belantara tersaji di pinggiran Kota Balikpapan, menikmati keindahan alam dengan cara yang sedikit ekstrim, yaitu dengan berpetualang menggunakan motor trail. Trail *adventure* atau enduro, saat ini menjadi salah satu olahraga yang di gemari. Mengingat olah raga ini memiliki kelebihan dari beberapa olahraga outbond lainya. Trail adventure juga telah merambah diberbagai kalangan tanpa mengenal batasan usia seperti pengusaha, pejabat daerah aparat polri, TNI, hingga pelajar.

Pengemar motor trail di Kota Balikpapan kerap melakukan turing bersama,berburu trek baru yang lebih menantang atau sekedar napak tilas jalur lawas. Komunitas tersebut adalah Komunitas Trail Adventure Balikpapan yang setiap sabtu atau minggu selalu menggelar event JELAS atau jelajah sabtu. Misi yang dibawa adalah menyatukan beberapa pencinta trail *adventure* di Kota minyak.

Jalur *offroad* yang populer bagi penggemar trail di Balikpapan adalah kawasan Syarifuddin Yoes dengan 3 cabang jalur, lajur kanan akan menuju sepinggan, lalu manggar dan berakhir dikawasan wisata pantai tanah merah. Lajur ke dua menuju ke jalan Sukarno-Hatta kilometer 8 dan jalur terakhir berada di kilometer 19 menuju taman wisata bukit bangkirai. Jalur ini penuh dengan tantangan dan rintangan, terutama lumpur yang tebal dan turunan curam.

Komunitas Trail Adventure Balikpapan, club yang diketuai oleh H. Arin Tafnida yang juga merupakan seorang pengusaha Billboard di Balikpapan ini sudah berdiri sejak tahun 2008 aktif menahkodai komunitas tersebut dan mengikuti event-event di seluruh penjuru kota di Indonesia. Saat ini Komunitas Trail Adventure Balikpapan mempunyai anggota sebanyak 175 anggota.

2.2. Mobile Ad-Hoc Networks (MANET*)*

Menurut Jun-Zhao Sun (2001), MANET adalah kumpulan *node* nirkabel yang dapat secara dinamis diatur di manapun dan kapanpun tanpa menggunakan infrastruktur jaringan yang sudah ada. MANET adalah sebuah sistem otonom di mana sebuah perangkat yang terhubung oleh *link* nirkabel bebas untuk bergerak secara acak dan bisa bertindak sebagai router atau *host* pada waktu yang sama. Jenis lalu lintas dalam jaringan *ad-hoc* sangat berbeda dari jaringan nirkabel dengan menggunakanan infrastruktur, yaitu :

a) *Peer-to-peer*

   Komunikasi antara dua node yang berada dalam 1-hop. *Network Traffic* (*bit per seconds*) biasanya bersifat konsisten.

b) *Remote-to-remote*

   Komunikasi antara dua *node* di luar 1-hop tetapi tiap *node* mempertahankan rute yang stabil di antara mereka. Hal ini merupakan hasil dari beberapa *node* yang tinggal dalam jangkauan komunikasi satu sama lain dalam satu area atau mungkin bergerak sebagai sebuah kelompok. Lalu lintas ini mirip dengan lalu lintas jaringan standar.

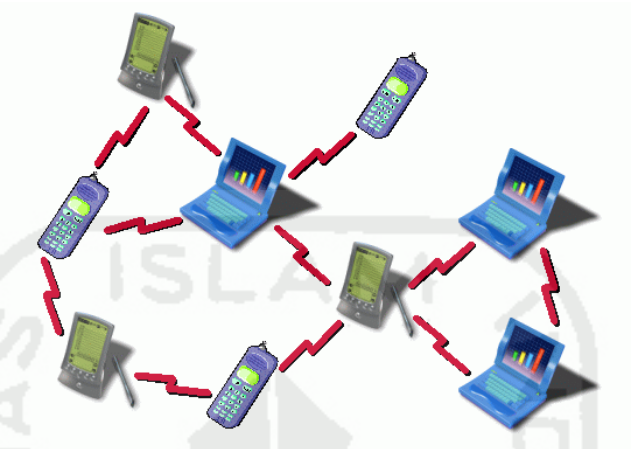
c) *Dynamic Traffic*

   Hal ini terjadi ketika ada sebuah *node* dinamis dan bergerak di *node* lain. Rute yang ada di dalam routing table harus dikonstruksi ulang untuk menyesuaikan penambahan jalur baru.

Gambar 2.1 Jaringan MANET

MANET terdiri dari *mobile platform* (seperti router dan perangkat *wireless*) dalam hal ini disebut dengan "*node*" yang bebas berpindah-pindah ke mana saja. *Node* tersebut bisa saja berada di pesawat, kapal, mobil dan dimana saja. Setiap *node* yang dilengkapi dengan *transmitter* dan *receiver wireless*

menggunakan. *Omnidirectional* berarti gelombang radio dipancarkan ke segala arah oleh perangkat *transmitter wireless*. Sedangkan *highly directional* adalah gelombang yang dipancarkan ke satu arah tertentu.



Gambar 2.2 Struktur Jaringan Nirkabel

MANET mempunyai beberapa fitur, antara lain:

a) *Autonomous Terminal.* Dalam MANET, setiap terminal *mobile* adalah suatu *autonomous node,* yang dapat berfungsi sebagai *host* dan *router.*

b) *Distributed operation.* Karena tidak ada jaringan yang bertindak sebagai pusat kontrol jaringan, kontrol dan manajemen jaringan didistribusikan di antara tiap terminal. Node yang termasuk di dalam MANET harus berkolaborasi dengan node lain untuk bertindak sebagai *relay* ketika dibutuhkan.

c) *Multihop Routing.* Jenis dasar algoritma *routing ad-hoc* dapat menjadi *single hop* dan *multi hop*, berdasarkan atribut *link layer* yang berbeda dan protokol *routing*. *Single hop* MANET lebih sederhana daripada *multi hop* dalam hal struktur dan implementasi, dengan biaya yang lebih rendah, fungsi, dan penerapan. Saat menyampaikan paket data dari sumber ke tujuan dari jangkauan transmisi nirkabel langsung, paket harus diteruskan melalui satu atau lebih node yang berada di tengah.

d) *Dynamic Network Topology*. Karena node bersifat *mobile*, topologi jaringan dapat berubah dengan cepat dan tak terduga dan konektivitas antara terminal

dapat bervariasi seiring dengan waktu. MANET harus beradaptasi dengan lalu lintas dan propagasi kondisi serta pola mobilitas *node* jaringan *mobile*. Node yang bersifat *mobile* dalam jaringan yang dinamis membangun dan membentuk routing dengan setiap *node* lainnya dengan cepat.

e) *Fluctuating link capacity*. Sifat tingginya *bit-error rates* dari koneksi jaringan nirkabel akan lebih sering terjadi di MANET, yaitu kondisi dimana suatu jalur *end-to-end* dapat digunakan oleh beberapa sesi.

f) *Lightweight Terminals*. Dalam kebanyakan kasus, node MANET adalah perangkat *mobile* dengan kapasitas kemampuan CPU yang tidak besar, ukuran memori kecil, dan penyimpanan daya rendah. Perangkat tersebut membutuhkan algoritma yang dapat mengoptimalkan sebuah mekanisme agar dapat melaksanakan fungsi komputasi dan komunikasi.

2.2.1. Sejarah dan Manfaat Mobile Ad-Hoc Network (MANET)

Sejarah perkembangan sistem jaringan *ad-hoc* dapat dikategorikan menjadi 3 generasi, yaitu generasi pertama, generasi kedua dan generasi ketiga. Sistem jaringan *ad-hoc* yang digunakan saat ini merupakan hasil dari *ad-hoc* generasi ketiga. Generasi pertama muncul pada tahun 1972, yang saat itu disebut dengan *Packet Radio Network* (PRNET). Jaringan *ad-hoc* awalnya merupakan proyek penelitian DoD1 yang disponsori oleh PRNET untuk tujuan militer pada tahun 1970-an, proyek tersebut kemudian berkembang menjadi program *Survivable Adaptive Radio Networks* (SURAN) pada awal tahun 1980. Hal-hal tersebut berkaitan dengan *Areal Locations of Hazardous Atmosphere* (ALOHA) dan *Carrier Sense Medium Access* (CSMA), seperti pendekatan untuk media akses kontrol dan jenis *routing distance vector* PRNET yang telah digunakan di setiap awal percobaan untuk memberikan kemampuan jaringan yang berbeda di dalam lingkungan perang.

Generasi kedua dari jaringan *ad-hoc* muncul di tahun 1980an, yaitu ketika sistem jaringan *ad-hoc* lebih ditingkatkan performanya dan dimasukkan sebagai bagian dari program *Survivable Adaptive Radio Networks* (SURAN). Program tersebut menyediakan jaringan yang dapat mengantarkan paket ke perangkat-

perangkat *mobile* di lingkungan tanpa infrastruktur. Program ini terbukti bermanfaat dalam meningkatkan kinerja radio dengan membuatnya menjadi lebih kecil, murah, dan tahan terhadap serangan elektronik.

Pada tahun 1990an konsep jaringan *ad-hoc* komersil muncul dengan komputer *notebook* dan peralatan komunikasi lainnya. Pada saat yang sama, ide dari sebuah koleksi *mobile node* telah diusulkan di beberapa konferensi para peneliti. Subkomite IEEE 802.11 telah mengadopsi istilah *ad-hoc networks* dan komunitas peneliti sudah mulai melihat kemungkinan penerapan jaringan Ad-hoc di area aplikasi lain. Aplikasi dari tipe tersebut memiliki jarak 1-hop jaringan *wireless* yang disebut *Wireless Local Area Network* (WLAN). Tipe kedua dari jaringan *wireless mobile* adalah jaringan *mobile* yang tidak menggunakan infastruktur atau disebut dengan MANET. MANET dapat mengorganisir dan mengatur dirinya sendiri dalam jaringan *multihop* yang tidak membutuhkan infrastruktur apapun untuk berkomunikasi. Di dalam sebuah jaringan, semua *node* bersifat dinamis dan dapat bergerak kemana saja, *node* tersebut dibutuhkan untuk menyampaikan paket ke *node* lain untuk mengirimkan data melalui jaringan.(Bang dan Prabhakar, 2013).

Keuntungan dari jaringan *Ad-Hoc* adalah sebagai berikut (Aarti & Tyagi, 2013):

1. Jaringan *Ad-hoc* dapat menyediakan akses untuk mencari informasi dan layanan yang tidak bergantung pada posisi geografis.
2. Tidak bergantung pada pusat administrasi jaringan. Jaringan *Ad-Hoc* juga dapat mengkonfigurasi jaringannya sendiri.
3. *Scalable*. Mengakomodasi jika ada penambahan *node* yang lebih dari satu.
4. Peningkatan flexibilitas.
5. Jaringan sangat kuat karena pusat administrasi jaringan tidak terpusat.
6. Jaringan dapat dibentuk di mana saja dan kapan saja.

2.2.2. Karakteristik MANET

Menurut Corson dan Macker (1999) MANET juga memiliki beberapa karakteristik, antara lain:

1. Topologi yang bersifat dinamis : *Node* pada MANET memiliki sifat yang

dinamis, yaitu dapat berpindah-pindah kemana saja. Maka topologi jaringan yang bentuknya adalah loncatan antara *hop* ke *hop* dapat berubah secara tidak menentu dan dapat terjadi secara terus menerus tanpa ada waktu yang pasti untuk berpindah. Topologi tersebut bisa saja terdiri dari *node* yang terhubung ke banyak *hop* lainnya, sehingga sangat berpengaruh secara signifikan terhadap susunan topologi jaringan.

2. *Node* mempunyai kemampuan otonomi: Setiap *node* pada MANET berperan sebagai *end-user* sekaligus sebagai router yang dapat menghitung sendiri *route-path* yang selanjutnya akan dipilih untuk mencapai tujuan tertentu.
3. Keterbatasan *bandwidth* : hubungan antar *node* pada jaringan *wireless* cenderung memiliki kapasitas yang rendah jika dibandingkan dengan jaringan yang menggunakan kabel. Jadi, kapasitas yang keluar untuk komunikasi *wireless* juga cenderung lebih kecil dari kapasitas maksimum untuk transmisi paket. Efek yang terjadi pada jaringan yang berkapasitas rendah adalah *congestion* (macet).
4. Keterbatasan energi : Semua *node* pada MANET bersifat *mobile*, sehingga setiap *node* membutuhkan energi yang cukup besar untuk terus beroperasi. Sehingga perlu suatu rancangan teknologi untuk mengoptimalkan energi.
5. Keterbatasan Keamanan : Jaringan *wireless* cenderung lebih rentan terhadap serangan dari luar jika dibandingkan dengan jaringan yang menggunakan kabel. Kegiatan pencurian (*eavesdroping*, *spoofing* dan *denial of service*) harus lebih diperhatikan.

2.2.3. Protokol MANET

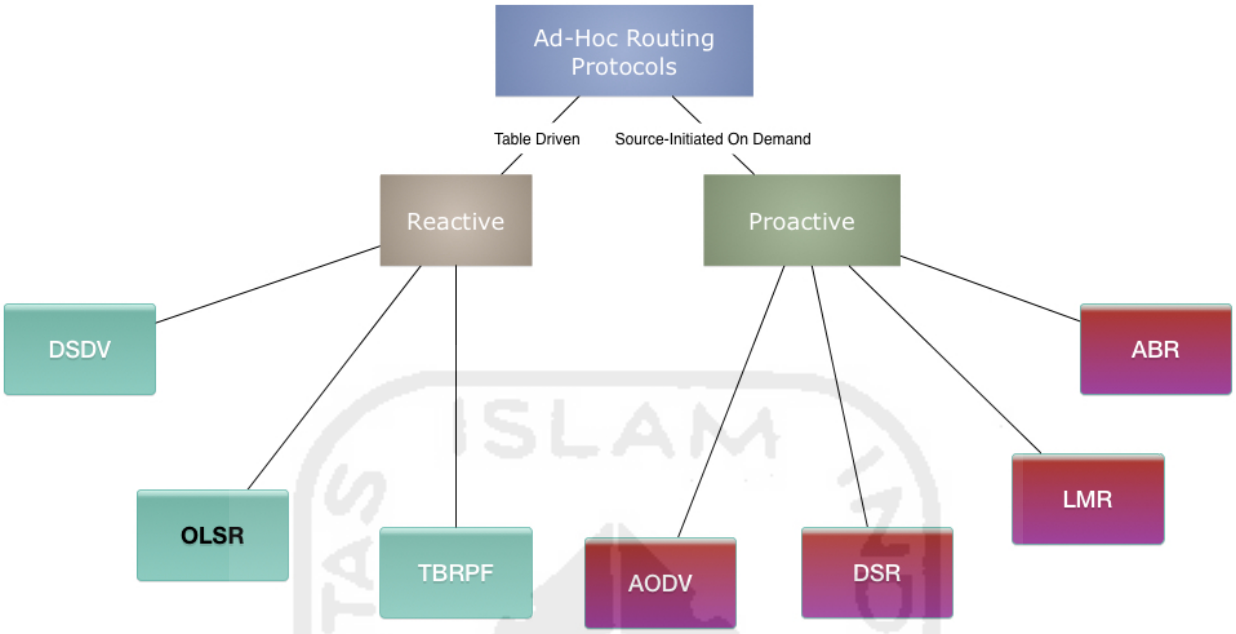
Protokol pada MANET terbagi menjadi dua, yaitu proaktif dan reaktif (Gambar 2.3). Routing proaktif mengelola daftar tujuan dan rute terbaru masing-masing *node* dengan cara mendistribusikan *routing table* ke seluruh jaringan, sehingga jalur lalu lintas (*traffic*) akan sering dilalui oleh *routing table* tersebut. Hal ini akan memperlambat aliran data jika terjadi restrukturisasi *routing table*. Sedangkan routing reactive akan mencari rute (*on demand*) dengan cara

membanjiri jaringan dengan paket *router request.* Sehingga dapat menyebabkan jaringan akan penuh (*clogging*).

Menurut Reddy, Raju, dan Venkatadri (2012), protokol proaktif berusaha untuk terus mengevaluasi rute dalam jaringan. Sehingga ketika sebuah paket harus diteruskan, rute tersebut sudah diketahui dan dapat segera digunakan. Protokol *distance vector* adalah contoh dari skema proaktif. Sedangkan protokol reaktif memanggil prosedur penentuan rute jika dibutuhkan saja. Dengan demikian, ketika suatu rute diperlukan, maka akan ada semacam prosedur pencarian secara global yang digunakan.

Menurut Shivahare, Wahi, dan Shalini (2012), setiap node dalam jaringan yang menggunakan *routing* reaktif memiliki tabel *routing* untuk disebarkan dari paket data ketika ingin membangun koneksi ke node lain dalam jaringan. Node-node tersebut menyimpan data mengenai semua tujuan yang dapat dijangkau dan jumlah *hop* yang diperlukan untuk tiba di tujuan masing-masing di dalam *routing* tabel. Entry *routing* sudah ditandai dengan nomor urut yang dibuat oleh node tujuan. Untuk mempertahankan stabilitas, tiap stasiun menyebar dan merubah tabel routing dari waktu ke waktu. Seperti berapa banyak hop yang diperlukan untuk tiba di node tertentu dan stasiun mana yang dapat dilewati. Setiap *node* yang menyebarkan data akan berisi nomor urutnya yang baru dan untuk setiap rute baru, node berisi informasi-informasi berikut:

1. Berapa banyak hop yang diperlukan untuk untuk tiba ke tujuan tertentu.
2. Generasi dari nomor urut yang baru yang ditandai dengan sampainya data ke node tertentu.
3. Alamat node tujuan.



Gambar 2.3 Protokol Routing MANET

Pada tugas akhir ini, protokol *routing* yang digunakan untuk simulasi dan saat penerapan MANET adalah OLSR.

2.2.4. Optimized Link State Protokol (OLSR)

Alasan pemilihan OLSR sebagai protokol *routing* adalah karena berdasarkan pengujian terdahulu yang dilakukan oleh Wahyu (2011), OLSR merupakan protokol yang lebih baik jika dibandingkan dengan AODV. Kesimpulan dari penelitian tersebut antara lain:

1. Hasil simulasi skenario 50 dan 100 *node* menunjukkan bahwa OLSR memiliki kinerja terbaik. Parameter kedua protokol yang mempengaruhi kinerja yaitu pesan *hello*. OLSR memiliki pengaturan pesan *hello* sebesar 2 detik sedangkan AODV sebesar 1 detik.
2. Berdasarkan hasil simulasi skenario 50 *node*, AODV memiliki *delay* sebesar 0,335 detik dan OLSR sebesar 0,079 detik. Pada simulasi skenario 100 *node,* AODV menghasilkan *delay* sebesar 1,186 detik dan OLSR sebesar 0,334 detik, sehingga OLSR lebih cepat dalam mencari rute. OLSR menggunakan jarak terpendek berdasar informasi tabel routing, sedangkan AODV menambahkan

jarak (hop) dari 1 *node* ke *node* lain.

3. Berdasarkan hasil simulasi skenario 50 *node*, AODV menghasilkan *load* sebesar 184.416.384 bit dan OLSR sebesar 191.866.476 bit. Pada simulasi skenario 100 *node,* AODV menghasilkan *load* sebesar 368.578.038 bit dan OLSR sebesar 330.149.010 bit. Hal ini menunjukkan AODV skenario 50 *node* lebih sedikit mengirim paket ke jaringan, sehingga AODV lebih cocok untuk jaringan skala kecil. Sebaliknya untuk skenario 100 *node*.

4. Berdasarkan hasil simulasi skenario 50 *node*, AODV memiliki *throughput* sebesar 841.199,38 bps dan OLSR sebesar 1.266.713,82 bps. Pada simulasi skenario 100 *node,* AODV memiliki *throughput* sebesar 068.832,52 bps dan OLSR sebesar 6.158.824,13 bps. Keberhasilan pengiriman paket lebih besar OLSR, karena informasi topologi OLSR selalu diperbarui jika ada perubahan terutama saat *node* bermobilitas. Setiap *node* dalam OLSR dapat menerima informasi topologi lebih dari sekali, sehingga menimbulkan pengulangan pesan *(overhead)*.

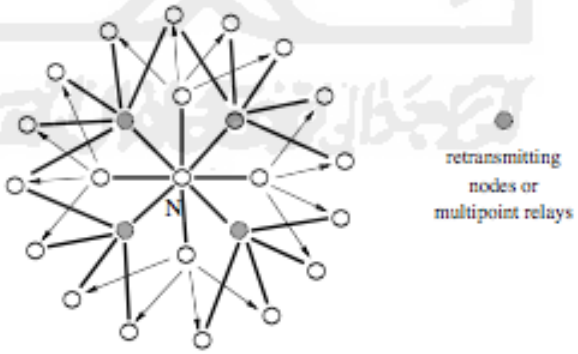
5. Berdasarkan hasil simulasi skenario 50 *node*, AODV menghasilkan PDR sebesar 96,02 % dan OLSR sebesar 97,80 %. Pada simulasi skenario 100 *node*, AODV menghasilkan PDR sebesar 76,40 % dan OLSR sebesar 95,67 %. OLSR lebih handal dalam mengirimkan paket jaringan terutama untuk jaringan dengan jumlah *node* lebih besar, karena OLSR menggunakan jarak terpendek dan tabel *routing* ketika merutekan paket, sedangkan AODV berdasarkan jarak *hop* antar tetangga.

Menurut Murthy dan Manoj (2004), *Optimized Link State Protokol* (OLSR) adalah sebuah protokol *routing* proaktif, jadi rutenya selalu secara cepat tersedia ketika dibutuhkan. OLSR adalah sebuah versi optimisasi dari sebuah protokol kondisi link murni (*pure link state protokol*). Perubahan secara topologi mengakibatkan luapan (*flooding*) informasi topologikal terhadap seluruh *node*/*host* yang berada di dalam jaringan. Untuk mengurangi jumlah *overhead* dalam jaringan digunakan sebuah teknik yaitu, dengan menggunakan teknik *Multi Point Relays* (MPR).

Tujuan dari *multipoint relays* adalah untuk meminimalisasi luapan

(*flooding*) dari *broadcast* paket-paket di jaringan dengan mengurangi salinan pengiriman kembali (*retransmission*) di daerah yang sama. Setiap node di dalam jaringan memilih kumpulan node disekitarnya, yang dapat mengirim ulang paketnya. Kumpulan node yang dipilih ini yang merupakan *multipoint relays* (MPR). Node-node lain yang tidak termasuk dalam MPR, dapat membaca dan memproses paket tetapi tidak dapat mengirim ulang paket yang disebar. Oleh karenanya, setiap node menyimpan informasi berupa kumpulan dari node-node disekitarnya yang disebut MPR *Selectors*. Setiap pesan *broadcast* yang datang dari MPR *Selectors* di asumsikan untuk dikirim ulang oleh node tersebut. Aturan ini dapat berubah seiring dengan waktu, yang terindikasi oleh *selector node* di dalam pesan HELLO mereka.

Setiap node memilih kumpulan *multipoint relay* nya sendiri diantara node yang berjarak 1-hop disekitarnya. *Multipoint Relay* yang diatur oleh node N, disebut MPR (N), adalah sebuah bagian yang bebas dari node-node sekitar dari *N* yang memenuhi beberapa kondisi: setiap node yang terletak dua hop di daerah sekitar node N harus memiliki hubungan dua arah menuju MPR (N). Bagian yang lebih kecil adalah sebuah kumpulan *multipoint relay*, sehingga routing protokol menjadi semakin optimal. **Gambar 2.4** menunjukkan pemilihan *multipoint relay* di sekitar node N.



Gambar 2.4 Pemilihan MPR Node N

Protokol OLSR bergantung pada pemilihan *multipoint relays*, dan menghitung rutenya ke semua tujuan yang diketahui melalui node-node tersebut.

Node MPR dipilih sebagai node yang berada di tengah rute. Untuk menerapkan skema ini, setiap node di dalam jaringan mengirim pesan *broadcast* tentang informasi 1-hop sekitarnya yang telah dipilih sebagai *multipoint relay.* Setelah menerima informasi *MPR Selector* ini, setiap node menghitung dan memperbarui rute ke setiap tujuan yang diketahui. Oleh karena itu, sebuah rute adalah urutan hop–hop melalui *multipoint relay* dari sumber ke tujuan (Clausen dan Jacquet, 2003).

2.2.4.1. Cara Kerja Protokol OLSR

Menurut Clausen dan Jacquet (2003), langkah-langkah kerja dalam OLSR dapat diurutkan sebagai berikut:

1. *Link Sensing* (Pendeteksian hubungan).
   Link Sensing dilakukan dengan mengirimkan pesan HELLO secara periodik dan berkesinambungan. Hasil dari *link sensing* adalah *local link set* yang menyimpan informasi hubungan antara *interface* yang ada pada *node* tersebut dengan *node-node* tetangga.
2. *Neighbour detection* (pendeteksian node tetangga).
   Node pengirim pesan HELLO akan menerima informasi alamat-alamat dari *node-node* tetangganya beserta *link status*-nya.
3. *MPR selection* (pemilihan MPR).
   Melalui pesan HELLO *node* utama akan menentukan sejumlah *node* tetangga untuk dipilih sebagai *multipoint relay* (MPR) yang bertugas meneruskan paket-paket kontrol ke dalam jaringan.
4. Pengiriman TC (*Topology Control*) *Messages.*
   TC *Messages* dikirimkan untuk memberikan informasi *routing* kepada setiap *node* yang ada pada jaringan yang akan digunakan untuk penentuan jalur.
5. *Route calculation (*penghitungan jalur).
   Berdasarkan informasi rute yang didapat dari paket-paket kontrol seperti

HELLO dan TC maka setiap *node* akan memiliki *routing table* yang berisi informasi rute yang dapat dilalui untuk dipakai mengirimkan data ke *node-node* lainnya yang ada pada jaringan.

2.2.4.1.1. Link Sensing

Setiap *node* pada jaringan dengan protokol OLSR harus mengetahui jenis hubungan yang dimiliki dengan *node-node* tetangganya. Jenis-jenis hubungan inilah yang kemudian digunakan untuk menentukan kedudukan *node-node* tetangga terhadap *node* tersebut. Proses pendeteksian hubungan dengan *node-node* tetangga tersebut dinamakan *Link Sensing* (Clausen and Jacquet, 2003).

Link sensing (pendeteksian hubungan) dikerjakan dengan pengiriman pesan HELLO secara periodik melalui *wireless interface* yang digunakan dalam *node* tersebut. Bila node tersebut menggunakan lebih dari satu *interface*, maka setiap *interface* akan mengirimkan HELLO *message* yang berbeda-beda.

Hasil pemrosesan data yang didapat dari HELLO *message* yang diterima oleh setiap *node* akan menghasilkan "*local link set*" yang berisi informasi tentang hubungan antara *local interface* (*interface* pada node tersebut) dengan *remote interface* (*interface* pada node tetangga). Informasi yang terdapat pada *link set* tersebut antara lain adalah :

- L_local_iface_addr,
- L_neighbor_iface,
- L_SYM_time,
- L_ASYM_time,
- L_time.

L_local_iface_addr merupakan alamat *interface* pada node tersebut, L_neighbor_iface adalah alamat *interface* pada node tetangga, L_SYM_time adalah waktu yang dibutuhkan hingga sebuah hubungan dianggap *simetris*, L_ASYM_time merupakan waktu yang dibutuhkan hingga *interface* tetangga dianggap terdeteksi, dan L_time merupakan waktu yang dibutuhkan hingga informasi ini dianggap kadaluarsa.

L_SYM_time digunakan untuk menentukan Link Type yang merupakan

status hubungan dengan *node-node* tetangga. Jika L_SYM_time tidak kadaluarsa, maka hubungan akan bersifat simetris. Sedangkan bila L_SYM_time telah kadaluarsa maka hubungan dengan *node* tetangga akan bersifat *asimetris*. Dan apabila L_SYM_time dan L_ASYM_time keduanya telah kadaluarsa maka hubungan dengan tetangga dinyatakan hilang (Clausen dan Jacquet, 2003).

2.2.4.1.2. Neighbour Sensing

Setiap *node* pada jaringan juga harus mendeteksi *node-node* tetangga yang ada pada daerah jangkauannya. Untuk melakukan hal tersebut, setiap *node* akan mengirimkan paket pesan HELLO secara *broadcast* dalam periode waktu tertentu. Paket HELLO berisi informasi tentang *node-node* tetangga serta *link status*.

Dalam setiap *node* akan menyimpan informasi tentang *node-node* tetangga tersebut dalam "*neighbor set*" . *Neighbor set* tersebut berisi informasi sebagai berikut :

- N_neighbor_main_addr
- N_status
- N_willingness

N_neighbor_main_addr merupakan alamat dari *node* tetangga, N_status adalah informasi yang menunjukkan apakah hubungan dengan *node* tetangga tersebut bersifat SYM atau NOT_SYM. N_willingness merupakan tingkat kesediaan *node* tersebut untuk meneruskan paket untuk kepentingan *node* lain yang ditunjukkan dalam bentuk integer 0-7. Selain *neighbor set* yang digunakan untuk menyimpan informasi tentang tetangga 1-*hop*, setiap node juga akan menyimpan informasi *node-node* tetangga 2-*hop* nya. Informasi tersebut disimpan dalam 2-*hop neighbor set* yang berisi informasi antara lain:

- N_neighbor_main_addr
- N_2hop_addr
- N_time

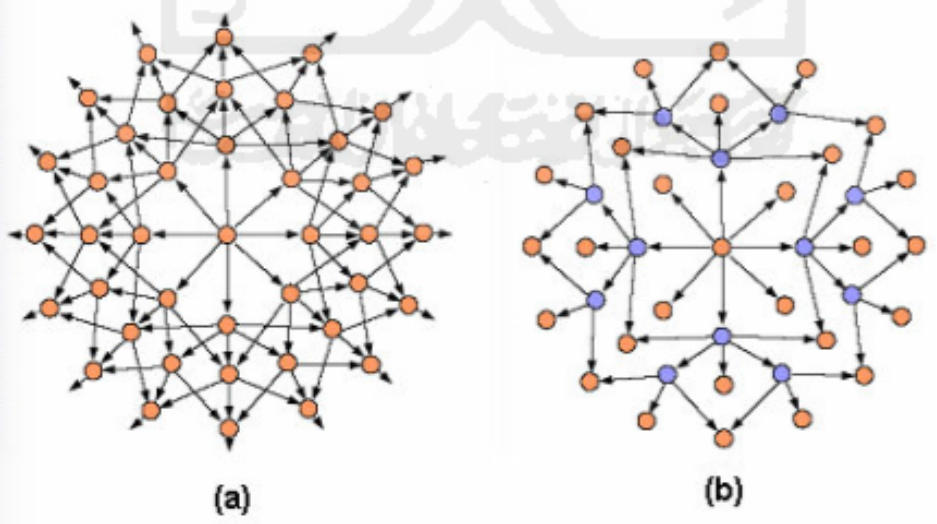
N_neighbor_main_addr merupakan informasi alamat dari *node* tetangga 1-*hop*, N_2hop_addr merupakan alamat dari *node* tetangga 2-*hop* yang bersifat simetris dengan tetangga 1-*hop* nya. N_time menunjukkan waktu yang dibutuhkan hingga

informasi yang ada pada set tersebut akan kadaluarsa dan harus dihilangkan (Clausen dan Jacquet, 2003).

2.2.4.1.3. MPR Selection

Tujuan dari penggunaan Multipoint Relay (MPR) adalah meminimalisir penggunaan *overhead* yang pesan broadcast pada jaringan dengan cara mengurangi retransmisi (pentransmisian ulang) pada daerah yang sama (Clausen and Jacquet, 2003). Setiap node pada jaringan akan memilih sejumlah node tetangga 1-hop nya yang bersifat simetris yang akan melakukan transmisi ulang pesan-pesannya. Sejumlah node tetangga tersebutlah yang disebut dengan MPR. Setiap node tetangga yang tidak terpilih menjadi MPR tetap akan menerima dan memproses pesan broadcast namun tidak akan meneruskan atau mengirimkan kembali pesan-pesan tersebut.

Pemilihan node-node untuk dijadikan MPR selain harus bersifat simetris juga harus sedemikian rupa dapat menjangkau sejumlah node tetangga 2-hop. Makin sedikit jumlah MPR maka makin sedikit penggunaan control traffic *overhead* yang digunakan dalam routing protocol. Perbandingan kinerja pengiriman paket untuk OLSR dan link state protocol pada umumnya digambarkan pada Gambar 2.5.



Gambar 2.5 Perbandingan Sistem Broadcast (a) General Broadcasting (b) MPR Broadcasting (Tonnesen, 2008)

Setiap *node* akan menyimpan informasi tentang *node-node* tetangga yang telah dipilihnya sebagi MPR dalam sebuah "*MPR set*" yang berisi alamat-alamat *node* MPR tersebut. Selain itu setiap *node* juga akan menyimpan informasi tentang siapa-siapa saja yang telah memilihnya sebagai MPR. Informasi tersebut disimpan dalam "MPR Selector Set" yang berisi informasi antara lain :

- MS_main_addr
- MS_time

Ms_main_addr merupakan alamat-alamat *node* tetangga yang telah memilihnya sebagai MPR. MS_time berisi informasi waktu yang dibutuhkan hingga informasi tersebut kadaluarsa dan harus dihilangkan (Clausen dan Jacquet, 2003). .

2.2.4.1.4. Topology Control Message Diffusion

Pendeteksian hubungan serta pendeteksian *node-node* tetangga dari protokol OLSR pada dasarnya menyediakan informasi daftar tetangga yang dapat berkomunikasi secara langsung, dan dikombinasikan dengan mekanisme *broadcast* dengan menggunakan MPR informasi topologi dapat dikirimkan ke seluruh jaringan.

Rute dibentuk dari *advertised link* dan hubungan dengan *node-node* tetangga. Setiap *node* setidaknya harus mempunyai informasi tentang hubungan antara dirinya sendiri dengan *node-node* yang ada pada *MPR-selector set* nya dalam rangka mendapatkan mendapatkan informasi routing yang baik.

Pesan TC dikirimkan untuk menyediakan informasi *link-state* bagi setiap *node* pada jaringan yang dapat digunakan untuk penentuan jalur yang dapat digunakan. Setiap *node* akan menyimpan informasi yang didapat dari pesan TC ini dalam "Topology Set" yang berisi informasi antara lain:

- T_dest_addr
- T_last_addr
- T_seq
- T_time

T_dest_addr merupakan alamat dari sebuah node yang dapat dicapai dalam 1-*hop* dari *node* dengan alamat yang pada T_last_addr. Sehingga T_last_addr

merupakan MPR dari *node* yang ada pada T_dest_addr. T_seq adalah nomor urut dan T_time menunjukkan batas waktu kadaluarsa dari informasi tersebut.

2.2.4.1.5. Routing Calculation

Dengan menggunakan informasi *link state* yang didapatkan dari pertukaran pesan secara periodik dan juga disertai dengan konfigurasi *interface* dari setiap *node* maka *routing table* dari setiap node dapat dikalkulasi.

Setiap *node* memiliki *routing table* yang dapat dipakai untuk jalur data menuju *node-node* lain dalam jaringan. *Routing* tersebut dibuat berdasarkan informasi dalam *local link information base* (*local link set, neighbour set, 2-hop neighbour set, MPR set*), serta informasi pada *topology set*. Oleh karena itu, apabila terjadi perubahan pada *set-set* tersebut maka *routing table* akan dikalkulasi ulang untuk meng-*update* informasi tentang setiap tujuan dalam jaringan.

Adapun informasi yang disimpan dalam suatu *routing table* adalah seperti terlihat pada tabel 2.1.

Tabel 2.1 Tabel Routing OLSR

1	R_dest_addr	R_next_addr	R_dist	R_iface_addr
2	R_dest_addr	R_next_addr	R_dist	R_iface_addr
3	.....	.....	.....	.....

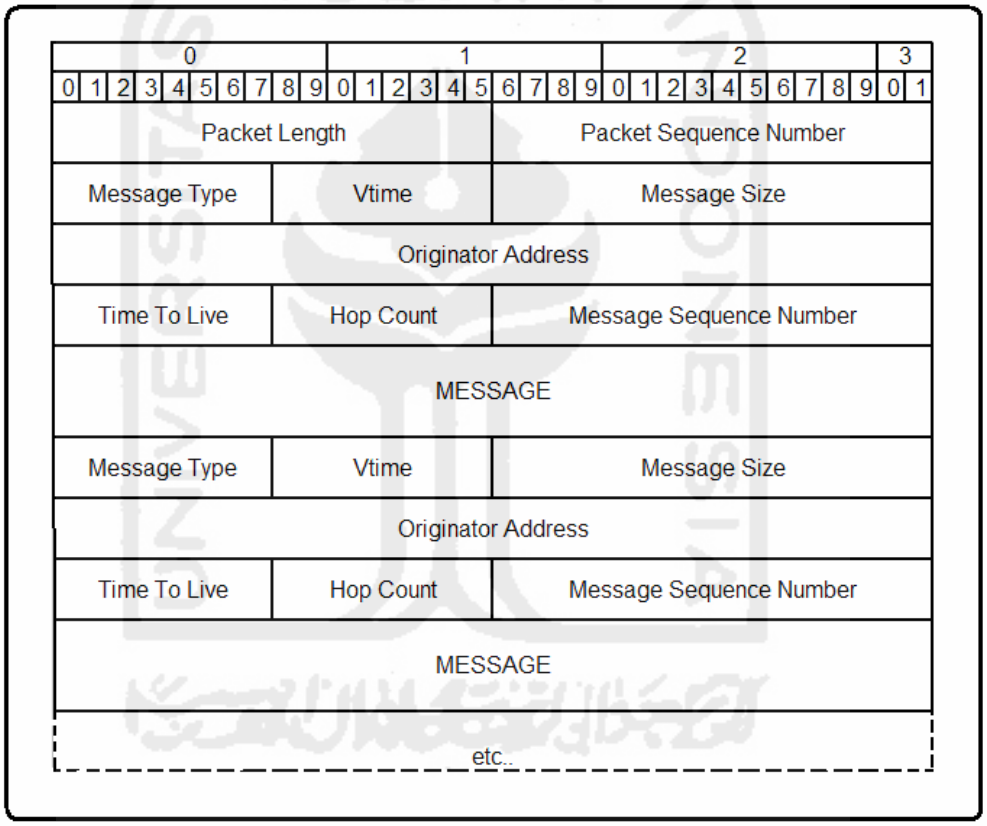
R_dest_addr menunjukkan alamat *node* yang dapat dituju sedangkan R_dist merupakan jarak atau jumlah *hop* yang harus dilalui untuk mencapai *node* tujuan tersebut. R_next_addr merupakan alamat node dari hop berikutnya yang dari rute untuk menuju alamat tujuan. R_iface_addr merupakan alamat *interface* pada *node* sumber yang dapat dipakai untuk menghubungi *node* pada R_next_addr.

Setiap *entries* yang disimpan dalam *routing table* untuk setiap *node* tujuan

yang ada pada jaringan dimana telah terdapat rute untuk menuju *node* tersebut. Setiap tujuan yang rutenya putus atau tidak memiliki informasi yang lengkap tidak akan dimasukkan dalam *routing table*.

2.2.4.2. Format Paket OLSR

Format paket pada protokol OLSR, paket tersebut dapat dibagi menjadi tiga bagian utama yaitu : *Packet Header, Message Header* dan *Message* (Gambar 2.6).



Gambar 2.6 Format Paket OLSR

a. *Packet Length,* panjang dari paket OLSR dalam satuan byte.
b. *Packet Sequence Number,* merupakan nomor urut dari pengiriman paket, *Packet Sequence Number* harus ditambah dengan nilai 1 setiap terjadi pengiriman paket yang baru.
c. *Message Type,* menampilkan jenis pesan dari paket tersebut.
d. *Vtime, Vtime* menjelaskan berapa lama waktu yang dibutuhkan untuk

menentukan validasi suatu paket oleh suatu *node* sejak diterimanya paket tersebut. *Vtime* dapat diperkirakan dengan menggunakan persamaan sebagai berikut:

Validity Time = C*($\frac{1+a}{16}$)*2^b [*in seconds*]

Dimana:

C = nilai dari scaling factor ($\frac{1}{16} detik$)

A = nilai integer dari empat bit pertama dari Vtime

B = nilai integer dari empat terakhir Vtime

e. *Message Size,* menunjukkan ukuran dari pesan tersebut dimuai, dinyatakan dalam satuan byte. Ukuran dari pesan dimualai dari awal *field Message Type* hingga bertemu awal dari *field Message Type* lainnya.

f. *Originator Address, field* ini berisi alamat utama dari node pengirim yang mengirimkan pesan tersebut untuk pertama kalinya.

g. *Time To Live, field* ini berisi jumlah maksimum dari hop yang dapat dilalui pada saat proses transmisi. Untuk setiap proses transimisi ulang, nilai dari *field* ini harus dikurangi dengan 1. Node yang menerima paket dengan nilai *Time to Live* sama dengan 0 atau 1 maka node tersebut tidak dapat meneruskan transmisi pesan tersebut.

h. *Hop Count, field* ini berisi informasi jumlah hop yang sudah dilewati oleh pesan tersebut. Nilai dari *field* ini harus dijumlahkan dengan nilai 1 setiap akan melakukan proses transmisi ulang.

i. *Message Sequence Number,* setiap terjadi pengiriman pesan, node pengirim akan memberikan nomor urut tertentu pada setiap pesan yang akan dikirimkan, fungsi dari pemberian nomor urut ini adalah untuk memastikan tidak terjadinya proses transmisi ulanh lebih dari satu oleh node manapun.

2.2.4.3. Keuntungan Menggunakan OLSR

Menurut Onno dan Surya (2013) keuntungan ketika menggunakan OLSR

adalah sebagai berikut:

a. *Overhead* atau waktu yang dibutuhkan untuk membuat *routing,* walaupun biasanya sedikit lebih lama jika dibandingkan protokol reaktif, tetapi tidak bertambah dengan jumlah rute yang dibuat.
b. Karena menggunakan protokol yang praoaktif, rute ke semua tujuan dalam jaringan akan diketehaui dan dipelihara sebelum digunakan. Karena rute akan tersedia dalam *routing* standar menjadikannya sangat berguna bagi beberapa sistem/aplikasi jaringan karena tidak ada waktu *delay* untuk mencari rute.
c. Rute *default* dan rute ke jaringan luar (*network*) dapat dimasukkan ke sistem menggunakan message HNA yang memungkinkan sambungan ke internet atau jaringan lain dalam OLSR MANET *cloud.* Rute *network* adalah salah satu kelebihan protokol proaktif yang saat ini tidak bisa ditangani oleh protokol rektif dengan baik.
d. Nilai *timeout* dan informasi validitas ada dalam *message* yang memungkinkan nilai *timer* yang berbeda digunakan untuk berbagai *node.*

2.3. Simulasi Jaringan

Menurut Teerawat dan Hossain (2009), sebuah simulasi dapat dianggap sebagai suatu proses aliran *entities* (entitas) jaringan (contoh: *node*, paket). Ketika entitas-entitas tersebut bergerak melalui sistem, mereka berinteraksi dengan entitas lain, bergabung dalam kegiatan tertentu, memicu peristiwa, menyebabkan beberapa perubahan keadaan pada sistem, dan meninggalkan proses. Dari waktu ke waktu, mereka bersaing atau menunggu untuk beberapa jenis sumber daya. Ini berarti bahwa harus ada urutan eksekusi logis untuk menyebabkan semua tindakan ini terjadi dengan cara yang dapat dipahami dan dikelola. Sebuah urutan eksekusi memainkan peran penting dalam mengawasi simulasi dan kadang-kadang digunakan untuk mencirikan jenis simulasi.

Dalam komunikasi dan penelitian jaringan komputer, simulasi jaringan adalah sebuah teknik di mana sebuah program memeragakan perilaku jaringan baik dengan menghitung interaksi antara entitas jaringan yang berbeda

(*host*/*router*, *link data*, paket, dll) dengan menggunakan rumus matematika, atau sebenarnya menangkap dan memutar kembali pengamatan dari produksi jaringan. Keuntungan utama dari simulasi jaringan adalah simulasi yang dilakukan tidak menyebabkan permasalahan atau bahkan membahayakan pada jaringan yang sesungguhnya atau setidaknya membutuhkan inisialisasi baru pada model element dan *traffic*. Oleh karena itu perilaku jaringan dan berbagai aplikasi dan layanan yang mendukung dapat diamati secara leluasa di laboratorium penguji; berbagai atribut lingkungan juga dapat dimodifikasi dengan cara yang terkontrol untuk menilai bagaimana jaringan akan berperilaku di bawah kondisi yang berbeda (Braun and Staub, 2008)

Menurut Teerawat dan Hossain (2009), komponen struktural simulasi terdiri dari:

1. *Entities*

   Entitas adalah objek yang berinteraksi dengan satu sama lain dalam sebuah program simulasi untuk menyebabkan beberapa perubahan pada keadaan dari sistem. Dalam konteks jaringan komputer, entitas mungkin termasuk *node* komputer, paket, aliran paket, atau objek non-fisik seperti jam simulasi. Untuk membedakan entitas yang berbeda, atribut yang unik ditugaskan untuk masing-masing entitas. Sebagai contoh, sebuah entitas paket mungkin memiliki atribut seperti panjang paket, nomor urut, prioritas, dan *header*.

2. *Resources*

   Sumber daya merupakan bagian dari sistem yang kompleks. Secara umum, persediaan sumber daya yang terbatas harus dibagi di antara kumpulan entitas tertentu. Hal ini biasanya terjadi untuk jaringan komputer, dimana *bandwidth*, *air time*, jumlah server, misalnya, mewakili sumber daya jaringan yang harus dibagi di antara entitas jaringan.

3. *Activities and Events*

   Dari waktu ke waktu, entitas terlibat dalam beberapa kegiatan. Keterlibatan akan hal ini menciptakan peristiwa dan memicu perubahan dalam keadaan

sistem. Contoh umum kegiatan meliputi *delay* dan *queueing*. Ketika komputer membutuhkan untuk mengirimkan paket tetapi menemukan medium sibuk, maka harus menunggu sampai medium bebas. Dalam hal ini, paket yang akan dikirim melalui udara tapi medium sibuk, paket dikatakan terlibat dalam aktivitas menunggu.

4. *Scheduler*

   Scheduler memelihara daftar kejadian dan waktu eksekusi mereka. Selama simulasi, *scheduler* menjalankan jam simulasi menciptakan peristiwa, dan mengeksekusi mereka.

5. *Global Varieties*

   Dalam simulasi, variabel global dapat diakses oleh fungsi atau entitas apa saja dalam sistem, dan pada dasarnya melacak beberapa nilai umum simulasi tersebut. Dalam konteks jaringan komputer, variabel seperti itu mungkin mewakili, misalnya, panjang dari antrian paket dalam jaringan server tunggal, total sibuk *air time* dari jaringan nirkabel, atau jumlah paket yang ditransmisikan.

6. *Random Number Generator*

   Sebuah *Random Number Generator* (RNG) diperlukan untuk memperkenalkan keacakan dalam model simulasi. Nomor acak dihasilkan oleh mengambil nomor secara berurutan dari urutan deterministik nomor *psudo-random*, namun nomor diambil dari urutan ini secara acak. Dalam kebanyakan kasus, urutan *psudo-random* telah ditetapkan dan digunakan oleh semua RNG. Dalam pelaksanaannya, RNG diinisialisasi dengan *seed*. *Seed* mengidentifikasi lokasi awal dalam urutan *psudo-random*, di mana sebuah RNG mulai memilih angka. Simulasi berbeda diinisialisasi dengan *seed* yang berbeda sehingga menghasilkan hasil yang berbeda (tapi secara statistik identik). Dalam simulasi jaringan komputer, misalnya, proses kedatangan paket, proses menunggu, dan proses layanan biasanya dimodelkan sebagai proses acak. Sebuah proses acak dinyatakan oleh urutan variabel acak. Proses acak ini biasanya dilaksanakan dengan bantuan dari

suatu RNG.

7. *Statistics Gatherer*

   Tanggung jawab utama dari seorang pengumpul statistik adalah untuk mengumpulkan data yang dihasilkan oleh simulasi sehingga kesimpulan yang berarti dapat ditarik dari data tersebut.

Pada tugas akhir ini, simulator yang digunakan untuk mensimulasikan pergerakan *node* sesuai dengan studi kasus yang ada adalah *Network Simulator 3*.

2.3.1. Network Simulator 3 (NS-3)

NS-3 adalah sebuah simulator jaringan peristiwa, target utama NS-3 adalah untuk kepentingan penelitian dan pendidikan. NS-3 adalah perangkat lunak gratis, berlisensi di bawah lisensi GNU GPLv2, dan tersedia untuk umum, penelitian, dan pengembangan.

Tujuan dari proyek NS-3 adalah untuk mengembangkan sebuah simulator yang disukai, sebuah ruang simulasi yang bersifat terbuka untuk penelitian jaringan. Hal ini harus disesuaikan dengan kebutuhan simulasi dari penelitian jaringan di era modern dan harus mendorong masyarakat luas untuk berkontribusi.

NS-3 mendukung *scheduler real-time* yang memfasilitasi sejumlah penggunaan kasus "*simulation-in-the-loop*" untuk berinteraksi dengan sistem yang nyata. Misalnya, pengguna dapat memancarkan dan menerima paket NS-3 yang dihasilkan pada perangkat jaringan yang nyata, dan NS-3 dapat berfungsi sebagai kerangka kerja interkoneksi untuk menambahkan efek hubungan antara mesin virtual (*http://nsnam.org*)

NS-3 merupakan sebuah simulator jaringan yang sering digunakan untuk simulasi protokol *routing* diantara yang simulator lainnya, dan juga sering digunakan untuk penelitian mengenai *ad-hoc networking*, dan mendukung protokol jaringan yang populer, serta menyediakan hasil simulasi untuk jaringan kabel maupun nirkabel. NS-3 juga cukup populer di kalangan peneliti karena berbasis *open source* serta menyediakan dokumentasi penelitian dari penelitian sebelumnya secara online pada website pengembang NS-3.

Istilah yang biasa terdapat pada *networking*, namun memiliki arti yang

spesifik pada NS-3, antara lain:

1. *Node*

   Dalam jargon internet, perangkat komputer yang terhubung ke jaringan disebut *host* atau terkadang *end-system*. Dalam NS-3 abstraksi perangkat komputasi dasar atau komputer disebut *node*. Abstraksi ini diwakili dalam C++ oleh kelas *Node*. Kelas *Node* menyediakan metode untuk mengelola representasi perangkat komputasi di simulasi.

2. *Application*

   Dalam NS-3 abstraksi dasar untuk program pengguna yang menghasilkan beberapa kegiatan yang akan disimulasikan adalah aplikasi. Abstraksi ini diwakili dalam C++ oleh kelas Application. Kelas Application menyediakan metode untuk mengelola representasi versi NS-3 pada aplikasi-aplikasi level user dalam simulasi. Pengembang diharapkan untuk mengkhususkan kelas Application dalam pengertian pemrograman berorientasi obyek untuk membuat aplikasi baru.

3. *Channel*

   Seringkali media dimana aliran data dalam jaringan disebut *channel*. Dalam dunia simulasi NS-3, seseorang menghubungkan sebuah *Node* ke objek yang mewakili sebuah saluran komunikasi. Di NS-3 abstraksi komunikasi dasar *subnetwork* disebut *channel* dan diwakili di C++ oleh kelas Channel.

4. *Net Device*

   Untuk terhubung dengan jaringan, komputer harus memiliki perangkat keras yang disebut dengan *peripheral card*. *Peripheral card* tersebut diimplementasikan beberapa fungsi jaringan, sehingga disebut *Network Interface Cards* (NICs). NIC tidak akan berfungsi tanpa sebuah *software driver* untuk mengontrol perangkat keras tersebut. Pada Unix (atau Linux), sebuah *peripheral hardware* disebut sebagai *device*. *Device* dikontrol menggunakan *device driver*, dan NIC dikontrol menggunakan *network device driver* yang disebut dengan *net device*. Di NS-3 *net device* meliputi baik *software driver* dan simulasi *hardware*. Sebuah *net device* ‘di-instalasi’ pada sebuah *Node* agar

memungkinkan *Node* untuk berkomunikasi dengan *Node* lainnya dengan simulasi melalui *Channels*. Abstraksi *net device* direpresentasikan dengan C++ oleh kelas *NetDevice*. Kelas *NetDevice* menyediakan metode untuk mengatur koneksi ke objek *Node* dan *Channel*.

5. *Topology Helpers* Dalam sebuah jaringan simulasi besar akan diperlukan banyak koneksi untuk mengatur antara *Node, NetDevice* serta *Channel*. NS-3 menyediakan apa yang disebut objek *Topology Helpers* untuk mengatur simulasi–simulasi jaringan semudah mungkin.

Jika dibandingkan dengan simulator jaringan lainnya, NS-3 dibedakan dengan tujuan desain tingkat tinggi berikut:

1. C++ dan Python penekanan

Banyak simulator menggunakan bahasa pemodelan domain-spesifik untuk menggambarkan model dan aliran program. NS-3 menggunakan C++ atau Python, yang memungkinkan pengguna untuk mengambil keuntungan dari dukungan penuh dari setiap bahasa.

2. *Callback-driven event* dan koneksi

Peristiwa simulasi di ns-3 adalah fungsi panggilan sederhana yang dijadwalkan untuk mengeksekusi pada waktu simulasi yang ditentukan. Setiap fungsi dapat dibuat menjadi sebuah acara dan dijadwalkan, dengan menggunakan fungsi *callback*. Hal ini berbeda dengan fungsi "*handler*" khusus yang memusatkan pengolahan peristiwa di setiap objek simulasi. *Callback* juga banyak digunakan dalam simulator untuk mengurangi ketergantungan kompilasi antara objek simulasi.

3. Lapisan inti yang fleksibel dengan lapisan *helper*

NS-3 memiliki tingkat API rendah yang kuat yang memungkinkan pengguna daya fleksibilitas untuk mengkonfigurasi hal-hal dengan cara yang berbeda. Lapisan di atas ini adalah satu set "pembantu" lapisan API yang menyediakan fungsi yang lebih mudah digunakan dengan perilaku *default* yang wajar. Pengguna NS-3 dapat mencampur dan mencocokkan antara API sederhana pada lapisan pembantu dan API penuh di bawahnya.

4. Penekanan pada emulasi

Desain simulasi berorientasi kasus penggunaan yang memungkinkan simulator untuk berinteraksi dengan dunia nyata. Paket objek NS-3 disimpan secara internal sebagai buffer packet byte (mirip dengan paket dalam sistem operasi nyata) siap untuk di serialisasi dan dikirim pada antarmuka jaringan nyata. Beberapa *simulation-in-the-loop* dan integrasi mesin virtual kerangka kerja yang berbeda telah dikembangkan, dan percobaan NS-3 telah dilakukan pada *testbeds* nirkabel.

5. Keselarasan dengan antarmuka dunia nyata

Node NS-3 berpola setelah arsitektur jaringan Linux, dan interface kunci dan benda-benda (soket, perangkat net) sejalan dengan yang di komputer Linux. Ini lebih baik memfasilitasi penggunaan kembali kode dan meningkatkan realisme dari model, dan membuat aliran kontrol simulator lebih mudah untuk membandingkan dengan sistem nyata.

6. Manajemen konfigurasi

Fitur simulator NS-3 sistem berbasis atribut terintegrasi untuk mengelola standar dan nilai-nilai per-contoh untuk parameter simulasi. Semua nilai default dikonfigurasi untuk parameter yang dikelola oleh sistem ini, terintegrasi dengan baris perintah pengolahan argumen, dokumentasi Doxygen, dan XML-based dan opsional subsistem konfigurasi berbasis GTK.

7. Kurangnya IDE

Proyek ini tidak mempertahankan *Integrated Development Environment* (IDE) untuk melakukan konfigurasi, *debug*, mengeksekusi, dan memvisualisasikan simulasi dijendela aplikasi tunggal, seperti yang ditemukan di simulator lainnya. Sebaliknya, alur kerja khas adalah untuk bekerja pada baris perintah dan mengintegrasikan alat konfigurasi dan visualisasi yang diperlukan. Beberapa pengembang telah menggunakan Eclipse sebagai alat untuk mengembangkan IDE.

2.4. Parameter Unjuk Kerja Jaringan

Menurut Gunawan (2008), berdasarkan sudut pandang jaringan, *Quality Of Service* (*QoS*) adalah kemampuan suatu elemen jaringan, seperti aplikasi jaringan, *host,* atau *router* untuk memiliki tingkatan jaminan bahwa elemen jaringan tersebut dapat memenuhi kebutuhan suatu layanan.

Adapun Parameter QoS yang digunakan dalam pengukuran adalah :

1. *Delay*

   Menurut Solekan (2009), *delay* adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kongesti atau juga waktu proses yang lama.

   Menurut Urida Z Ilma (2011), semakin besar nilai *delay* maka akan semakin menurunkan performa dari suatu jaringan, karena itu nilai *delay* harus seminimum mungkin. Menurut Urida Z Ilma (dalam versi Tiphon), kategori jaringan di bagi menjadi 4 bagian *delay* yaitu:

   Tabel 2.2 Kategori jaringan berdasarkan nilai delay (versi Tiphon)

Kategori	**Nilai *Delay***
Sangat Baik	0 ms
Baik	75 ms
Buruk	125 ms
Sangat Buruk	225 ms

2. *Throughput*

   Menurut Oppenheimer (2011), *bandwidth* adalah kemampuan sebuah jaringan untuk membawa data, biasanya diukur dalam bit per detik (bps). Konsep *bandwidth* tidak cukup untuk menjelaskan kecepatan jaringan dan apa yang terjadi di jaringan. Untuk itulah konsep *Throughput* muncul. *Throughput*, yaitu kecepatan (*rate*) transfer data efektif, yang diukur dalam bps. *Troughput* merupakan jumlah total kedatangan paket yang sukses yang diamati pada *destination* selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut (Solekan, 2009) . Hal ini sama dengan, jumlah pengiriman paket IP sukses per *service-second*.

Menurut Urida Z Ilma (2011), persamaan yang di gunakan untuk mencari throughput adalah :

$$throughput = \frac{\text{jumlah data yang dikirim}}{\text{waktu pengiriman}}$$

3. *Packet Loss*

Menurut Solekan (2009), *packet loss* merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang, dapat terjadi karena *collision* dan *congestion* pada jaringan dan hal ini berpengaruh pada semua aplikasi karena *retransmisi* akan mengurangi efisiensi jaringan secara keseluruhan meskipun jumlah *bandwidth* cukup tersedia untuk aplikasi-aplikasi tersebut. Salah satu penyebab *packet loss* adalah antrian yang melebihi kapasitas *buffer* pada setiap *node*. Beberapa penyebab terjadinya *packet loss* yaitu:

- *Congestion*, disebabkan terjadinya antrian yang berlebihan dalam jaringan
- *Node* yang bekerja melebihi kapasitas *buffer*
- *Memory* yang terbatas pada *node* *Policing* atau kontrol terhadap jaringan untuk memastikan bahwa jumlah trafik yang mengalir sesuai dengan besarnya *bandwidth*. Jika besarnya trafik yang mengalir didalam jaringan melebihi dari kapasitas *bandwidth* yang ada maka *policing control* akan membuang kelebihan trafik yang ada.

Tabel 2.3 Kategori jaringan berdasarkan nilai *packet loss* (versi tiphon)

Kategori	**Nilai *packet loss***
Sangat Baik	0%
Baik	3%
Buruk	15%
Sangat Buruk	25%