BAB II

LANDASAN TEORI

2.1 Mobile Ad Hoc Network (MANET)

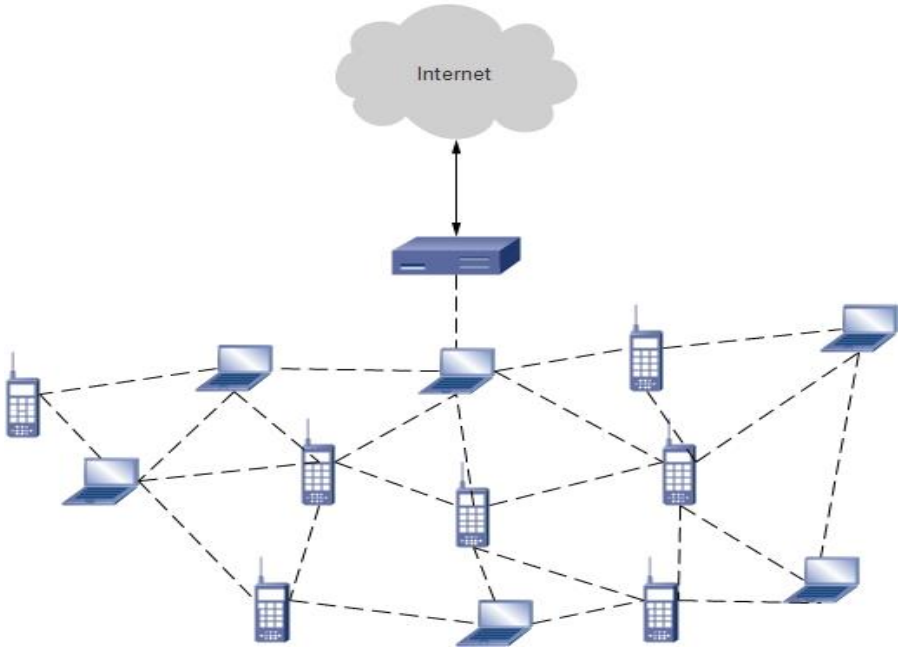
MANET adalah *autonomous system* dari *node* mobile yang terhubung secara nirkabel. Setiap *node* beroperasi tidak hanya sebagai *end system*, tetapi juga sebagai router untuk meneruskan paket. *Node-node* tersebut bebas untuk bergerak dan mengorganisir diri dalam sebuah jaringan. *Node-node* tersebut saling berkolaborasi dengan cara saling memforwarding *packet* satu dengan lain sehingga membuat *node-node* tersebut dapat berkomunikasi pada area outside dari *direct wireless transmission*. MANET tidak membutuhkan *centralized administration* ataupun *fixed network infrastructure* seperti *base station* atau *access points* (Thakare AN, Joshi MY, 2012).

Setiap *node* dilengkapi dengan *transmitter* dan *receiver wireless* menggunakan antenna atau sejenisnya yang bersifat *omnidirectional* (*broadcast*), *highly directional* (*point to point*), memungkinkan untuk diarahkan, atau dikombinasi dari beberapa hal tersebut. *Omnidirectional* maksudnya adalah gelombang radio dipancarkan ke segala arah oleh perangkat transmitter wireless. Sedangkan highly directional adalah gelombang yang dipancarkan ke satu arah tertentu (Rudhyanto PF, 2016).

Topologi jaringan MANET dapat berubah dengan cepat dan tak terduga dari waktu ke waktu, karena *node* bersifat mobile. Tipe *network* dalam MANET adalah *desentralisasi*, dimana semua aktivitas jaringan, termasuk menemukan topologi dan penyampaian pesan harus dijalankan oleh *node* itu sendiri. Oleh karena itu fungsi *routing* harus dimasukkan ke dalam *node mobile* (Thakare AN, Joshi MY, 2012).

MANET menjadi topik penelitian di tahun 1990-an saat laptop dan *wi-fi* mulai populer. Secara umum penggunaan MANET yang pernah diterapkan adalah pada kemiliteran untuk kebutuhan pemeliharaan komunikasi antara tentara militer dengan kendaraan dan markas pusat komunikasi. Penggunaan MANET sendiri

sampai sekarang masih menjadi bahan penelitian karena aspek penggunaannya yang masih luas. Pada gambar 2.1 menunjukkan jaringan MANET sederhana :



Gambar 2.1 MANET (Abdalla., 2008).

Mobilitas host bisa menyebabkan sering terjadinya perubahan topologi yang tidak bisa diprediksikan, karena itu tugas menemukan dan mempertahankan rute adalah salah satu tugas penting protokol penentuan rute *ad hoc*. Protokol penentuan rute digunakan di dalam jaringan *ad hoc* yang berbeda dalam menemukan dan mempertahankan rute baru di dalam jaringan. Parameter yang mempengaruhi kinerja protokol penentuan rute jaringan *ad hoc* adalah beberapa *mobile* di dalam area, perilaku *mobile* yang mengubah *konektivitas* (bergerak secara bebas sebagai kelompok, frekuensi perubahan topologi), dan kualitas *link*. Pada MANET terdapat beberapa keuntungan, beberapa diantaranya adalah (Purba DU, Primananda R, Amron K, 2017):

- Tidak memerlukan dukungan *backbone* infrastruktur, sehingga dapat dengan mudah diimplementasikan dan sangat berguna ketika infrastruktur tidak ada ataupun tidak dapat berfungsi lagi.

- *Mobile node* yang selalu bergerak atau dinamis dapat mengakses informasi secara *real time* ketika berhubungan dengan *mobile node* lainnya, sehingga pertukaran data dan pengambilan keputusan dapat segera dilakukan.
- Fleksibel, karena jaringan ini memang memiliki sifat yang sementara.
- Dapat direkonfigurasikan dalam bermacam-macam topologi, baik untuk jumlah *user* kecil hingga jumlah *user* besar sesuai dengan aplikasi dan instalasi (*scalability*)

Sedangkan kekurangannya adalah :

- *Packet loss* akan terjadi bila transmisi mengalami kesalahan (*error*).
- Seringkali terjadi *disconnection*, karena tidak selalu berada dalam area cakupan (*network area*).
- *Bandwith* komunikasi terbatas.
- *Lifetime* baterai yang singkat.

2.1.1 Karakteristik jaringan *Ad Hoc*

Seperti yang telah diketahui bahwa *node-node* yang ada pada jaringan MANET tidak hanya berperan sebagai *node* pengirim dan penerima saja, tetapi juga berfungsi sebagai router yang mampu menentukan rute untuk pengiriman data. Beberapa karakteristik MANET diantaranya (Harahap EH., 2014):

a. Topologi yang 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 terpola dan terjadi secara terus menerus tanpa ada ketetapan waktu untuk berpindah. Bisa saja didalam topologi tersebut terdiri dari *node* yang terhubung ke banyak hop lainnya, sehingga sangat berpengaruh secara signifikan terhadap susunan topologi jaringan.

b. Otonomi : Setiap *node* pada MANET berperan sebagai *end-user* sekaligus sebagai router yang menghitung sendiri *route-path* yang selanjutnya akan dipilih.
c. Keterbatasan *bandwidth* : *Link* pada jaringan *wireless* cenderung memiliki kapasitas yang rendah jika dibandingkan dengan jaringan berkabel. Jadi, kapasitas yang keluar untuk komunikasi *wireless* juga cenderung lebih kecil dari kapasitas maksimum transmisi. Efek yang terjadi pada jaringan yang berkapasitas rendah adalah *congestion* (kemacetan).
d. Keterbatasan energi : Semua *node* pada MANET bersifat *mobile*, sehingga sangat dipastikan *node* tersebut menggunakan tenaga baterai untuk beroperasi. Sehingga perlu perancangan untuk optimalisasi energi.
e. Keterbatasan Keamanan : Jaringan *wireless* cenderung lebih rentan terhadap keamanan daripada jaringan berkabel. Kegiatan pencurian (*eavesdroping*, *spoofing* dan *denial of service*) harus lebih diperhatikan.

2.1.2 Jenis-jenis *routing* protokol pada MANET

Routing merupakan suatu mekanisme penentuan jalur komunikasi dari *node* pengirim ke *node* penerima. *Routing* bekerja pada layer ketiga OSI (layer network). Untuk melakukan pengiriman data (informasi) tersebut, maka protokol *routing* akan bertugas untuk menentukan jalur yang akan digunakan untuk mengirimkan data sampai tiba ditempat penerima (Harahap EH, 2014).

Terdapat berbagai jenis *routing protocol* untuk MANET yang secara keseluruhan dapat dibagi menjadi beberapa kelompok, antara lain (Yanuar GC, 2016):

a. Proactive *Routing*

   Algoritma ini akan mengelola daftar tujuan dan rute terbaru masing-masing 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*. Beberapa contoh algoritma proactive *routing* adalah :

- Babel
- B.A.T.M.A.N (Better Approach to Mobile Ad hoc Network)
- DSDV (Highly Dynamic Destination Sequenced Distance Vector *routing* protocol)
- HSR (Hierarchial State *Routing* Protocol)
- IARP (Intrazone *Routing* Protocol)
- LCA (Linked Cluster Architecture)
- WAR (Witness Aided *Routing*)
- OLSR (Optimized *Link* State *Routing* Protocol)

b. Reactive *Routing*

Tipe ini akan mencari rute (*on demand*) dengan cara membanjiri jaringan dengan paket *router request.* Sehingga dapat menyebabkan jaringan akan penuh (*clogging*). Beberapa contoh algoritma reactive *routing* adalah :

- SENCAST
- Reliable Ad Hoc On Demand Distance Vector *Routing* Protocol
- Ant-Based Control Enabled On Demand *Routing* (ACOR)
- Ariadne
- Associativity Base *routing*
- Ad Hoc On Demand Distance Vector (AODV)
- Ad Hoc On Demand Multipath Distance Vector (AOMDV)
- Backup Source *Routing*
- Dynamic Source *routing* (DSR)
- Flow State in the Dynamic Source *Routing*
- Dynamic MANET On Demand *Routing* (DYMO)

c. Flow Oriented *Routing*

Tipe protokol ini mencari rute dengan mengikuti aliran yang disediakan. Salah satu pilihan adalah dengan unicast secara terus menerus ketika meneruskan data saat mempromosikan *link* baru. Beberapa kekurangan tipe protokol ini adalah membutuhkan waktu yang lama untuk mencari rute baru. Beberapa protokol yang memiliki tipe ini adalah :

- Interzone *Routing* Protokol (IERP)
- Lightweight Underlay Network Ad Hoc *Routing* (LUNAR)
- Signal Stability *Routing* (SSR)

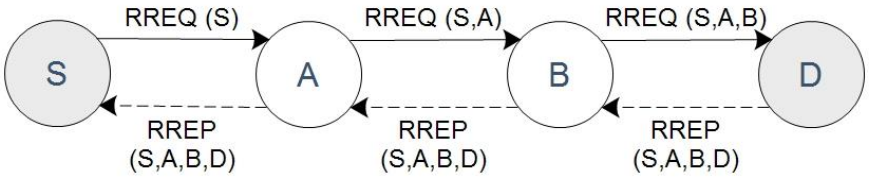
d. Hybrid *Routing*

Tipe protokol ini menggabungkan antara *proactive routing* dengan *reactive routing.* Protokol untuk tipe ini adalah :

- Hybrid *routing* protocol for large scale MANET (HRPLS)
- Hybrid wireless mesh protocol (HWMP)
- Zone *routing* protocol (ZRP)

2.2 Ad Hoc On Demand Distance Vector (AODV)

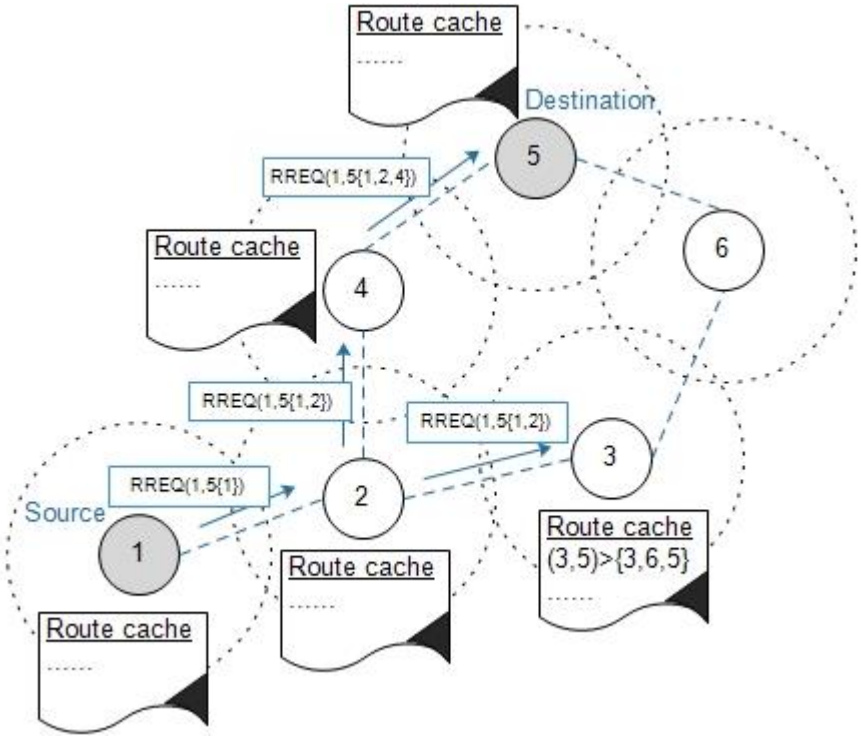
AODV merupakan protokol *routing* yang bersifat reaktif. Protokol ini bersifat reaktif karena protokol ini mulai bekerja saat ada permintaan dari *source node* untuk mencari tahu jalur-jalur yang akan digunakan untuk mengirimkan pesan ke *node* tujuan. AODV akan berusaha untuk menemukan jalur yang tidak ada *loop* dan menemukan jalur terpendek untuk menuju *node* tujuan (Harahap E. H., 2014). Terdapat dua pesan dalam proses pembentukan rute atau *route discovery process* pada *routing* AODV yaitu dengan menggunakan *route request* (RREQ) dan *route reply* (RREP). Pada proses *route maintenance* terdapat satu pesan yaitu *route error* (RRER) yang dimana apabila terjadi rute eror maka *node* akan menyebarkan pesan tersebut sampai ke *node* sumber dan akan melakukan *route discovery* kembali.



Gambar 2.2 Mekanisme Protokol AODV

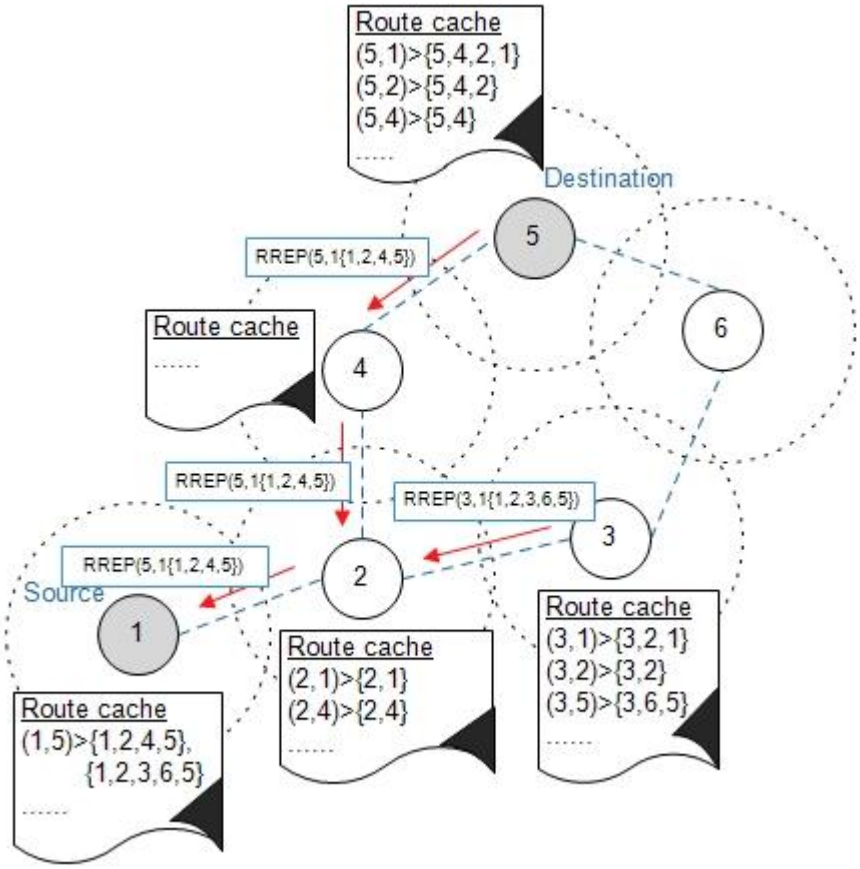
2.2.1 Proses *Route Discovery* Pada Protokol AODV

Route discovery adalah suatu mekanisme pada protokol AODV yang mempunya fungsi dalam pencarian rute secara dinamis dalam jaringan *ad hoc*. Penentuan rute ini terbagi menjadi dua bagian yaitu *route request* (RREQ) dan *route reply* (RREP). Jika *node* sumber menginginkan suatu rute menuju *node* tujuan tetapi masih belum ada rute menuju *node* tujuan maka *node* sumber akan menginisialisasi *route discovery process* untuk menemukan rute ke *node* tujuan dengan langkah-langkah berikut (Arinatal YA, 2015):



Gambar 2.3 Route Request (Fahriani N, Djanali S, Shiddiqi AM, 2012)

Pada gambar 2.3 *Node* (sumber) 1 (RREQ(1,5,{1})) mengirimkan route request terhadap *node* tetangga 2 (RREQ(1,5,{1})). *Node* pada *hop* berikutnya yaitu 2 (RREQ(1,5{1,2})) mengirimkan route request pada *node* 3 (RREQ(1,5{1,2})) dan *node* 4 (RREQ(1,5{1,2})), selanjutnya pada jangkauan range transmisi terdekat *nodei* 4 mengirimkan route request terhadap *node* 5 (RREQ(1,5,{1,2,4})). *Node* 5 sebagai destination address (*node* tujuan).



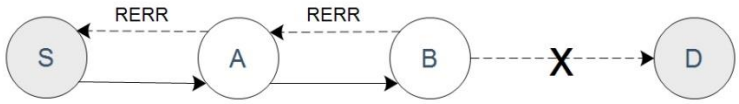
Gambar 2.4 Route Reply (Fahriani, N., dkk, 2012)

Pada gambar 2.4 destination (*node* tujuan) melakukan RREP terhadap *node* 4 (RREP (5,1{1,2,4,5})) *route cache* (5,1) > {5,4,2,1}, (5,2) > {5,4,2}, (5,4) > {5,4}. *Node* 4 melakukan RREP kepada *node* 2 (RREP(5,1{1,2,4,5})) *route cache* (2,1) > {2,1}, (2,4) > {2,4}. Begitu pula dengan *node* 3 juga mengirimkan RREP kepada *node* 2 (RREP(3,1{1,2,3,6,5} *route cache* (3,1) > {3,2,1}, (3,2) > {3,2}, (3,5) > {3,6,5}. Dan *route reply* akan berakhir pada

node 1 (RREP(5,1{1,2,4,5})) *route cache* (1,5) > {1,2,4,5}, {1,2,3,6,5} karena pada *node* tersebut yang melakukan proses awal RREQ.

2.2.2 Proses *Route Maintenance* Pada Protokol AODV

Route maintenance adalah sebuah proses yang bergantung pada perubahan topologi. Untuk menjaga rute. Setiap *node* akan mencoba untuk mendeteksi kesalahan koneksi (ketika ada *node* yang keluar dari jangkauan, atau beberapa peristiwa yang membatasi sebuah komunikasi) secara terus menerus. *Node* mendengarkan pesan RREQ dan RREP untuk melakukan ini. Selain itu, setiap *node* melakukan perjanjian untuk mengirim pesan setiap detik. Jika tidak ada RREQ atau RREP yang dikirim selama periode itu maka pesan Hello akan di kirim untuk menunjukkan bahwa *node* tersebut masih hidup atau tidak. Cara lain adalah mekanisme lapisan *link-layer* dapat digunakan untuk mendeteksi kesalahan koneksi. Selain melakukan pengamatan sebuah kesalahan koneksi, sebuah node juga harus merespon ketika menerima data paket yang tidak memiliki rute tujuan. Apabila terjadi *route error* (RERR), maka akan melakukan *route discovery* ulang (Sugianto, D. R., 2013).



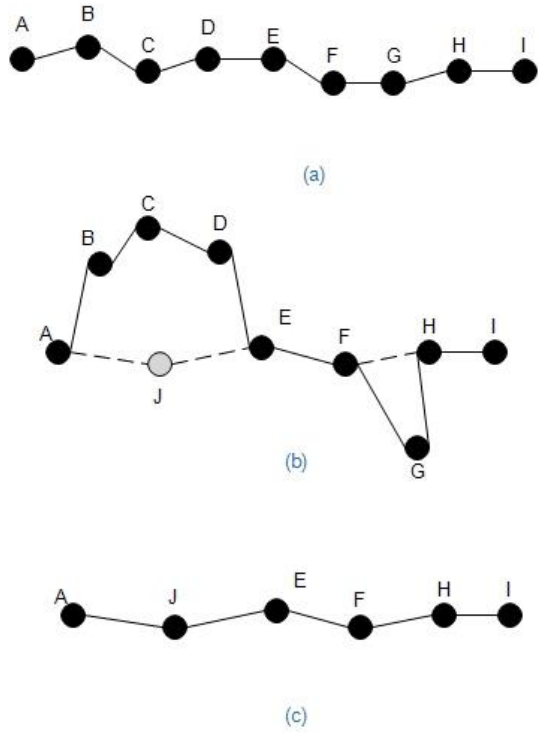
Gambar 2.5 Route Error (Wahanani, H.E. 2013)

Pada Gambar 2.5, jika *node* B tidak menerima pengakuan dari *node* tujuan D setelah beberapa permintaan karena terjadi kegagalan jalur, ia mengembalikan paket RERR ke *node* sumber S. Begitu *node* sumber S menerima pesan RERR, maka akan menghapus jalur rute yang rusak dari *cache*. *Node* sumber S kemudian mengulangi proses *route discovery*. Dalam protokol AODV, setiap kali jalur *node* terdeteksi mengalami kegagalan, rute yang ada terputus, maka *node* sumber harus memulai proses *route discovery*. Dalam topologi yang sangat dinamis kegagalan *node* sering terjadi, yang

menghasilkan proses *route discovery* yang menghabiskan waktu jika *node* B jauh dari *node* sumber S yang tidak memiliki rute alternatif dimana mengakibatkan *delay*.

2.3 Path-Aware Short

Jalur yang dihasilkan protokol pengaturan rute yang dibutuhkan dapat menyimpang jauh dari jalur terpendek. Karena mobilitas *node-node*, bentuk jalur pengaturan rute dapat secara signifikan saat konektivitas utuh. Perubahan bentuk dapat dieksploitasi untuk mendapatkan jalur-jalur pengaturan rute yang lebih baik jika dapat menghindari setiap *overhead* yang signifikan (setidaknya menghindari proses-proses penemuan *path* ekstra).



Gambar 2.6 Perubahan jalur pengaturan rute (Fahriani N, Djanali S, Shiddiqi AM, 2012).

Perhatikan jalur pengaturan rute dari *node* sumber A ke*node* tujuan I seperti yang ditunjukkan Gambar 2.6 (a). jalur awal ini ditentukan melalui proses penemuan jalur, dimana jarak antara sumber dan tujuan adalah jarak terpendek menurut jumlah *hop*. Paket membutuhkan 8 *hop* saat bergerak dari A ke I. selama

perjalanan waktu, mobilitas *node* dapat membuat bentuk jalur pengaturan rute menjadi seperti Gambar 2.6 (b) saat berusaha mempertahankan konektivitas. Dalam bentuk baru ini, J ada dalam tingkatan transmisi F. penggunaan *cache* rute dan validitas informasitentang pengaturan rute pada entri tabel pengaturan rute tidak diupdate. Idealnya, jalur terpendek dari A ke H hanya membutuhkan *hop* seperti yang ditunjukkan Gambar 2.6 (c).

2.4 Parameter Pengukuran

Berikut adalah beberapa parameter yang digunakan dalam menguji kinerja PA-AODV terhadap AODV yaitu (Fahriani N, Djanali S, Shiddiqi AM):

1. *Energy*

   Energy adalah *energy* yang digunakan pada saat pengiriman paket antara *node* pengirim ke *node* tujuan.
2. *Average end-to-end delay*

   Average end-to-end delay adalah jeda waktu antara paket pertama dikirim dengan paket tersebut diterima.
3. *Normalized routing load*

   Normalized routing load adalah waktu yang dibutuhkan antara jumlah paket yang dikirim dengan jumlah paket yang diterima.

2.5 Penelitian Sebelumnya

Beberapa penilitian sebelumnya tentang *routing* protokol AODV sebagai berikut:

1. Penilitan yang dilakukan oleh (Windianto W, Djanali S, Husni M, 2015) yang berjudul "*Optimasi routing pada protokol AODV_EXT dengan menggunakan Link Expiration Time (LET)"*. Pada penilitian ini *routing* protokol AODV sudah dimodifikasi dengan algoritma EXT, yang dimana saat RREQ dikirimkan maka berdasarkan kondisi tertentu hanya beberapa *intermediate node* yang akan meneruskan paket. *Intermediate node* yang telah menerima RREQ *message* akan memeriksa apakah densitas atau jumlah *node* tetangga memnuhi persyaratan. Apabila memenuhi, maka *intermediate node* tersebut akan

meneruskan paket. Protokol ini diberi nama AODV_EXT. Pada protokol AODV_EXT dilakukan penilitian optimasi rute dengan menggunakan algoritma Link Expiration Time (LET) dimana parameter yang akan diuji adalah packet delivery ratio (PDR), troughput, number of packet dropped, dan average end-to-end delay dengan parameter simulasi jumlah *node* 10, 20, 30, 40, 50, 100, 150 dan 200. Hasil simulasi yang didapat dengan menggunakan *Network Simulator 2* adalah algoritma *Link Expiration Time* pada AODV_EXT dapat meningkatkan kinerja protokol *routing* dalam pengiriman data. Protokol R-AODV_EXT menunjukkan kinerja terbaik dibandingkan AODV_EXT untuk parameter PDR dan Troughput pada kondisi jumlah *node* 100. Sedangkan kinerja terbaik untuk paramter *number of packet drop* berada pada kondisi jumlah *node* 50. Pada semua kondisi *Routing Overhead* pada R-AODV_EXT adalah lebih rendah daripada AODV_EXT tapi selisih nilainya kurang signifikan. Khusus pada jumlah *node* 10 nilai *Routing Overhead* untuk R-AODV_EXT adalah lebih tinggi dari pada AODV_EXT. Nilai *Average end-to-end delay* pada R-AODV_EXT cenderung lebih rendah dibanding AODV_EXT walaupun pada beberapa total *node* nilainya lebih tinggi.

2. Penilitian yang dilakukan oleh (Harahap EH, 2014) yang berjudul "*Analisis performansi protokol AODV (Ad hoc on Demand Ditance Vector) dan DSR (Dynamic Source routing) terhadap Active Attack pada MANET (Mobile Ad hoc Network) ditinjau dari QoS (Quality of Service) jaringan*". Pada penilitian ini dibandingkan kinerja performansi dua protokol *routing* MANET yaitu AODV dan DSR. Kedua protokol ini diberikan active attack. Active attack yang diberikan adalah *rushing attack, sinkhole attack, replay attack,* dan *sybil attack*. Parameter yang digunakan adalah *packet delivery ratio, average delay, average troughput,* dan *routing overhead*. Parameter simulasi yang digunakan yaitu jumlah *node* 10, 15, dan 20 dengan kecepatan masing masing *node* 15, 20, dan 25 m/s. Hasil simulasi yang

didapat dengan menggunakan *Network Simulator 2* adalah *active attack* menurunkan performansi jaringan dengan penurunan *packet delivery ratio* terbesar yaitu 16,4242% terjadi pada *sybil attack* menggunakan protokol AODV dengan jumlah *node* 15 dan kecepatan 25 m/s, penurunan *average delay* paling besar terjadi pada *rushing attack* dengan menggunakan AODV pada 20 *node* dan kecepatan 15 m/s sebesar 2968,3354% ms, penurunan *troughput* terbesar terjadi pada *sybil attack* pada protokol DSR sebesar 5.4949 Kbps di 20 *node* dengan kecepatan 20 m/s dan penurunan *routing overhead* paling besar terjadi pada *replay attack* dengan protokol AODV sebesar 243.1667% dengan 15 *node* pada kecepatan 25 m/s. Karena danya penurunan performansi akibat *active attack*, maka penganan untuk *rushing attack* terbaik menggunakan protokol DSR dengan jumlah *node* 20 dan kecepatan 25 m/s karena penurunan troughput sebesar 0,648%, *sinkhole attack* dapat dihadapi dengan menggunakan protokol DSR dengan 10 *node* dan kecepatan 15 m/s karena dapat mempertahankan *packet delivery ratio*nya, *replay attack* dapat dihadapi dengan menggunakan protokol DSR tanpa adanya penurunan *delay* dengan kecepatan 20 m/s pada 20 *node* dengan tanpa penurunan *routing overhead*.

3. Penilitian yang dilakukan oleh (Permana MY, Purwanto Y, Wahida I, 2010) yang berjudul "*Analisis pengaruh penggunaan protokol routing AODV, DSDV, dan ZRP pada performansi jaringan ad hoc Hibrid*". Pada penilitian ini disimulasikan tiga *routing* protokol yaitu AODV, DSDV, dan ZRP menggunakan Network Simulator 2. Tujuannya adalah mengevaluasi kinerja dari ketiga *routing* protokol tersebut pada jaringan ad hoc hybrid terhadap jumlah node dan koneksi serta peningkatan mobilitas. Parameter yang digunakan adalah rata-rata end-to-end delay, packet deliveri ratio, rata-rata troughput, dan routing overhead. Hasil simulasi yang di dapat yaitu Dari 2 macam skenario yang disimulasikan, disimpulkan ZRP memiliki kinerja yang lebih baik

dibandingkan dengan dua routing protocol lainnya. Pertama dilihat dari prosentase packet delivery ratio ZRP yang selalu diatas 98%. Kedua, ZRP memiliki nilai rata-rata throughput selalu lebih tinggi yaitu selalu bernilai sekitar 2 kali lipat daripada AODV dan DSDV, kecuali pada saat 125 koneksi yang hanya sampai sekitar 1,6 kali lipat. Dan ketiga, selalu memiliki rata-rata end to end delay yang lebih kecil daripada DSDV maupun AODV dengan perbedaan sekitar 30 ms untuk skenario koneksi, dan sekitar 15 ms untuk skenario mobilitas. Kinerja dari AODV lebih baik daripada ZRP dan DSDV terjadi pada routing overhead, dimana AODV selalu memiliki presentase yang lebih kecil yaitu sekitar 10% dari total paket. Dilihat dari analisa kinerja ketiga routing protocol terhadap penambahan jumlah node dan koneksi, protokol routing ZRP lebih unggul dibanding protokol routing DSDV dan AODV. Hal ini berarti protokol routing hybrid lebih unggul dalam manghadapi peningkatan kepadatan traffic dibandingkan protokol yang bersifat proaktif dan reaktif. Dalam analisa kinerja ketiga routing protocol terhadap peningkatan mobilitas, protokol routing DSDV dan ZRP lebih unggul dibanding protokol routing AODV. Dapat disimpulkan bahwa protokol yang bersifat reaktif tidak unggul pada jaringan yang banyak terjadi perubahan pada topologinya, yaitu saat mobilitas tinggi.

4. Penilitian yang dilakukan oleh (Purba DU, Primananda R, Amron K, 2017) yang berjudul "*Analisis Kinerja Protokol Ad Hoc on-Demand Distance vector (AODV) dan Fisheye State Routing (FSR) pada Mobile Ad Hoc Network*". Dalam penelitian ini, protokol routing yang digunakan yaitu AODV dan FSR. Protokol routing AODV membentuk sebuah rute dari satu node sumber ke node tujuan berdasarkan pada permintaan node sumber tersebut. Protokol routing FSR, setiap node menyimpan tabel yang berisi informasi rute pada setiap node yang diketahuinya, informasi rute akan diperbaharui secara berkala jika terjadi perubahan link. Parameter kinerja routing protokol yang diukur

berupa *packet delivery ratio, end to end delay*, *throughput* dan *packet loss* dengan menggunakan Network Simulator 2.35. parameter simulasi dengan jumlah *node* 20, 30, 40, dan 50. Hasil simulasi yang di dapat dengan menggunakan Network Simulator 2.35 adalah performansi routing protokol FSR unggul pada nilai parameter throughput dan end to end delay dengan nilai rata-rata throughput 108.435 kbps dan end to end delay 16.06575 m/s. Sedangkan, routing protokol AODV unggul pada nilai packet delivery ratio dan packet loss dengan nilai rata-rata packet delivery ratio sebesar 98.95 % dan packet loss 1.05 %. Pada skenario penambahan ukuran paket data, performansi dari routing protokol FSR juga unggul berdasarkan nilai parameter kerja throughput pada ukuran paket data 512 bytes dan end to end delay pada ukuran paket data 1024 bytes. Hal ini disebabkan semakin besar ukuran paket data, maka semakin besar waktu pengukuran yang dibutuhkan dalam suatu jaringan. Kondisi seperti ini yang dapat mempengaruhi hasil peforma dari throughput. Sedangkan, routing protokol AODV unggul pada packet delivery ratio dengan ukuran paket data 512 bytes dan packet loss dengan ukuran paket data 512 bytes. Hal ini disebabkan semakin besar ukuran paket data, maka waktu transfer yang dibutuhkan PDR semakin lama. Kondisi seperti ini juga mempengaruhi nilai packet loss pada saat melakukan pengiriman paket data menuju node tujuan, sehingga memungkinkan paket data yang hilang semakin meningkat. Performansi routing protokol berdasarkan nilai kinerja Quality of Service (QoS) terhadap penambahan jumlah node dan variasi ukuran paket sangat berpengaruh pada saat melakukan pengiriman paket data. Hasil yang diperoleh dalam pengujian, dapat disimpulkan bahwa protokol AODV lebih baik pada saat melakukan pengiriman paket data dibandingkan dengan protokol FSR. Secara keseluruhan, protokol AODV dan protokol FSR memiliki tingkat keunggulannya masingmasing. Protokol FSR cocok digunakan untuk kepadatan jaringan berskala kecil. Sedangkan,

protokol AODV lebih cocok digunakan untuk kepadatan jaringan dengan skala menengah atas. Hal ini disebabkan karena penambahan node sangat berpengaruh pada penurunan nilai kinerja parameter QoS. Semakin besar kepadatan node maka semakin banyak hop yang terbentuk sehingga dapat menghambat proses pengiriman paket data menuju node tujuan.

5. Penilitian yang dilakukan oleh (Putranto ATS, 2016) yang berjudul "*Analisis Penggunaan Energi AODV dan DSDV pada Mobile Ad Hoc Network*". Pada penelitian ini di bandingkan penggunaan energi pada *routing* protokol AODV dan DSDV dengan menggunakan Network Simulator 2. Parameter yang digunakan adalah *death of node* dan *troughput* dengan parameter simulasi pada area 300m x 300m, 800m x 800m, 1200m x 1200m, pada kecepatan 10, 20, dan 30 m/s, dan pada *node* simulasi penambahan 25, 75, dan 100. Hasil yang didapat pada *routing* protokol AODV cukup baik dalam menghemat baterai baik dalam penambahan area, kecepatan dan node karena AODV bersifat reaktif sehingga *node* yang penggunaan daya paling boros adalah sourcenya. Akan tetapi pada node yang sedikit dan kecepatan rendah AODV akan terlihat lebih banyak mengkonsumsi daya dikarenakan node sedikit akan selalu menjadi relay dan kecepatan rendah node masih dapat mempertahankan sebuah rute. Kemudian pada area yang kecil AODV terlihat cukup hemat dikarenakan rute yang digunakan pendek atau hop yang digunakan kecil. Pada *routing* protokol DSDV sangat boros baterai baik dalam penambahan area, kecepatan dan node dikarenakan DSDV bersifat proaktif untuk menjaga sebuah topologi jaringan dan rute *node* dengan cara melakukan update *route* pada semua node secara periodik. Akan tetapi pada area yang kecil dan besar serta kecepatan rendah dan tinggi DSDV cukup boros dalam penggunaan daya dikarenakan masih dapat mempertahankan sebuah route table. Kemudian pada *node* yang semakin bertambah DSDV lebih hemat dikarenakan *node* yang banyak melakukan control

message daripada mengirimkan packet data. Pada *routing* AODV memiliki *troughput* yang baik pada area yang sangat kecil dan penambahan *node*, akan tetapi pada kecepatan yang tinggi AODV mengalami penurunan yang signifikan dikarenakan link antar *node* akan selalu putus sehingga lebih sering melakukan control message dari pada mengirimkan packet data. Pada *routing* DSDV memiliki *troughput* yang baik pada area yang cukup besar dan kecepatan tinggi, akan tetapi pada penambahan *node* mengalami penurunan yang signifikan dikarenakan topologi yang sangat padat DSDV mengalami control *routing* yang tinggi.

2.6 Sejarah Network Simulator

Network simulator (NS) pertama kali dibangun sebagai varian dari real network simulator pada tahun 1989 di UCB (University of California Berkeley) pada tahun 1995, sekelompok tim gabungan membangun sebuah perangkat lunak simulasi jaringan internet untuk kepentingan riset interaksi antar protokol dalam kontes pengembangan protokol internet pada saat itu maupun masa yang akan datang.

2.6.1 Network Simulator 2

Network Simulator 2 (NS-2) dibuat untuk membantu menjalankan event-event yang dibuat pada penelitian di bidang jaringan (*networking).* Network Simulator menyediakan pendukung substansial untuk melakukan simulasi TCP, *routing* dan *multicast protocol* baik pada jaringan kabel maupun *wireless* (secara lokal maupun dengan satelit).

2.6.2 Kelebihan NS-2

Kelebihan dari NS-2 yaitu sebagai perangkat lunak simulasi pembantu analisis dalam riset atau penelitian. NS-2 dilengkapi dengan *tool* validasi. *Tool* validasi digunakan untuk menguji validitas pemodelan yang ada pada NS-2. Pembuatan simulasi dengan menggunakan NS-2 jauh lebih mudah daripada menggunakan *software developer* lainnya. Pada software

NS-2 ini *user* tinggal membuat topologi dan skenario simulasi yang sesuai dengan riset anda. Pemodelan media, protokol dan network component lengkap dengan perilaku trafiknya sudah tersedia pada library NS-2. NS-2 bersifat *open source* di bawah GPL (*Gnu Public License*), sehingga NS-2 dapat didownload melalui *website* NS-2 http://www.isi.edu/nsnam/dist.

2.6.3 Simulasi yang Menggunakan NS-2

NS-2 mensimulasikan jaringan berbasis TCP/IP dengan berbagai macam medianya. Anda dapat mensimulasikan protokol jaringan (TCPs/UDP/RTP), Traffic behaviour (FTP, Telnet, CBR, dan lain – lain), Queue management (RED, FIFO, CBQ) algoritma *routing* unicast (Distance Vector, *Link* State) dan multicast, (PIM SM, PIM DM, DVMRP, Shared Tree dan Bi directional Shared Tree), aplikasi multimedia yang berupa *layered video*, *Quality of Service videoaudio dan transconding.* NS-2 juga mengimplementasikan beberapa MAC (IEEE 802.3, 802.11), di berbagai media misalnya jaringan kabel (seperti LAN, WAN, *point to point*), nirkabel (seperti *mobile* IP, *Wireless* LAN), bahkan simulasi hubungan antar *node* jaringan yang menggunakan media satelit.

2.6.4 Konsep Dasar NS-2

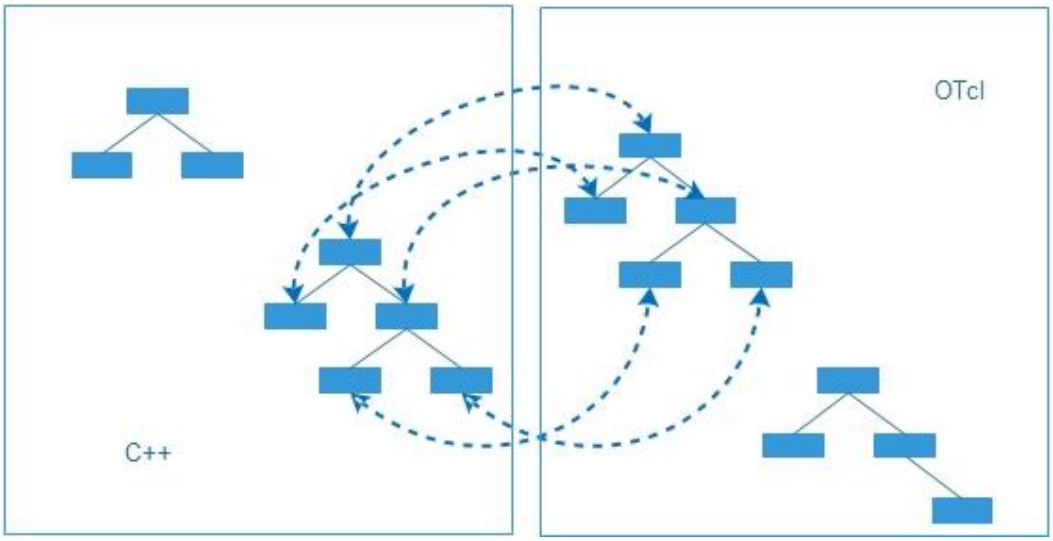
Network Simulator merupakan salah satu perangkat lunak atau software yang dapat menampilkan secara simulasi proses komunikasi dan bagaimana proses komunikasi tersebut berlangsung. *Network Simulator* melayani simulasi untuk komunikasi dengan kabel dan komunikasi nirkabel. Pada Network Simulator terdapat tampilan atau display baik dengan *node* yang bergerak atau *node* yang tidak bergerak, yang tentunya tidak sama dengan keadaan yang sebenarnya. Paket - paket yang membangun dalam simulasi jaringan ini antara lain :

- *Tcl : Tool Command Language*
- *Tk : Tool Kit*
- *Otcl : Object Tool Command Language*
- *Tclcl : Tool Command Language / C++ Interface*

• *NS-2 : Network Simulator versi 2*

• *Nam : Network Animator*

Network Simulator dibangun dengan menggunakan 2 bahasa pemrograman, yaitu C++ dan Tcl/Otcl. C++ digunakan untuk library yang berisi event scheduler, protokol dan network component yang diimplementasikan pada simulasi oleh user. Tcl/OTcl digunakan pada script simulasi yang ditulis oleh NS user dan pada library sebagai simulator objek. OTcl juga nantinya berperan sebagai interpreter. Hubungan antar bahasa pemrograman dapat dideskripsikan seperti Gambar 2.6.



Gambar 2.7 Hubungan C++ dan Otcl

2.6.5 Dasar Bahasa TCL dan OTCL

Tcl atau yang lebih dikenal dengan *Tool Command Language* adalah bahasa pemrograman yang didasarkan pada string atau *string – based command*. Tcl di desain untuk menjadi 'perekat' dalam membangun software *building block* untuk menjadi suatu aplikasi. Sedangkan OTCL (Object Oriented Tcl) adalah ekstensi tambahan pada Tcl yang memungkinkan fungsi *object oriented*. Hal ini memungkinkan dalam pendefinisian dan penggunaan class Otcl.

2.6.6 Cara Membuat dan Menjalankan Script NS

Script simulasi dibuat dengan menggunakan program teks editor pada OS yang digunakan, dan disimpan dalam sebuah *folder* dengan ekstensi.*tcl*, misalnya simulasi.*tcl*. Untuk menjalankan simulasi yang telah anda buat, anda tinggal masuk ke dalam folder tersebut dan mengetikkan NS serta nama file tcl simulasi yang ingin dijalankan.

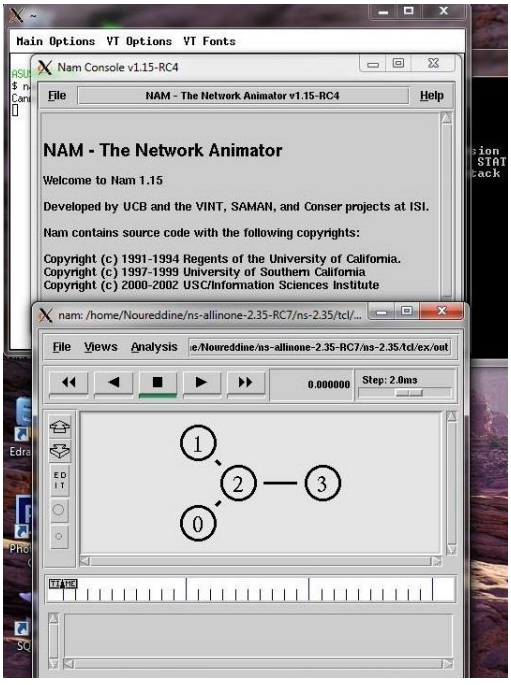
Contoh : [root@accessnet your_folder]#ns simulasi.tcl

2.6.7 Output Simulasi NS-2

Pada saat satu simulasi berakhir, NS membuat satu atau lebih *file output text based* yang berisi detail simulasi jika dideklarasikan pada saat membangun simulasi. Ada dua jenis output NS, yaitu :

• *File trace* : Digunakan untuk analisa numerik

• *File namtrace* : Digunakan sebagai input tampilan grafis simulasi yang disebut *network animator* (nam) yang dapat dilihat pada Gambar 2.5.



Gambar 2.8 NAM Console.