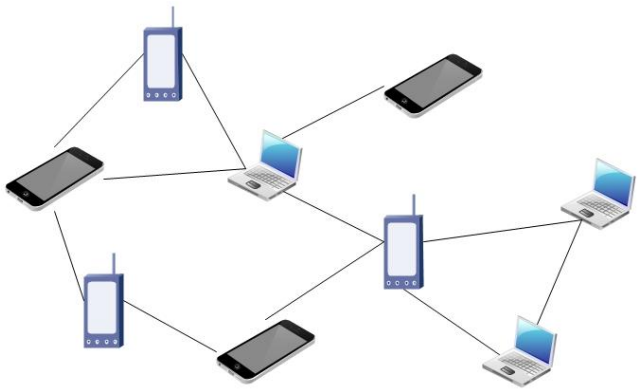
BAB II

LANDASAN TEORI

2.1 Mobile Ad-Hoc Network (MANET)

Mobile Ad Hoc Network (MANET) adalah sebuah jaringan tanpa kabel yang terdiri atas *mobile node* yang bergerak secara acak. *Node-node* dalam jaringan ini berfungsi juga sebagai *router* yang bertanggung jawab untuk mencari dan menangani rute ke setiap *node* didalam jaringan. *Node* bergerak bebas secara acak, dengan demikian *topologi* di jaringan mungkin dapat berubah dengan cepat dan tidak dapat diprediksi. Untuk mengatasi pergerakan ini diperlukan suatu protokol *routing* yang digunakan untuk menentukan rute antar *node* agar setiap *node* dapat berkomunikasi dan bertukar informasi (Rudhyanto, P.F. 2016).

Setiap *node* dilengkapi dengan *transmitter* dan *receiver wireless* menggunakan antena 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, P.F. 2016). Pada gambar 2.1 menunjukkan contoh jaringan MANET sederhana.



Gambar 2.1 *Mobile Ad hoc Network* (Ms.S.Suganya., dkk. 2012)

Mobile Ad Hoc Network (MANET) memiliki kelebihan antara lain (Rudhyanto, P.F. 2016) :

- Tidak memerlukan dukungan infrastruktur sehingga mudah diimplementasikan dan sangat berguna ketika infrastruktur tidak ada ataupun tidak berfungsi lagi.
- *Mobile node* yang selalu bergerak dapat mengakses informasi secara *real time* ketika berhubungan dengan *mobile node* lain, sehingga pertukaran data dan pengambilan keputusan dapat segera dilaksanakan.
- Fleksibel terhadap suatu keperluan tertentu karena jaringan ini memang bersifat sementara.
- Dapat direkonfigurasi dalam beragam topologi baik untuk jumlah *user* kecil hingga besar sesuai dengan kebutuhan.

Sedangkan kekurangannya adalah :

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

2.1.1 Karakteristik MANET

Berdasarkan dokumen *Request for Comments* menjelaskan bahwa terdapat beberapa karakteristik dari *Mobile Ad Hoc Network* (MANET). Disana dijelaskan bahwa 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. *Mobile Ad Hoc Network* (MANET) juga memiliki beberapa karakteristik yang lebih menonjol, antara lain (Corson, S. dkk, 1999) :

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.

2.2 Protokol Routing Pada MANET

Routing merupakan suatu mekanisme penentuan jalur komunikasi yang menghubungkan dari *node* pengirim ke *node* penerima. Untuk melakukan pengiriman data (informasi) tersebut, maka protokol *routing* akan bertugas

untuk menentukan jalur yang akan digunakan untuk mengirimkan data sampai tiba di tempat penerima (Anggraini, S.D., dkk. 2017).

Dalam jaringan *ad hoc*, setiap *node* tidak mempunyai pengetahuan mengenai topologi jaringan sekitarnya, melainkan bahwa *node-node* harus dicari. Dasar pemikirannya adalah bahwa *node* baru yang masuk akan mendengarkan pesan *broadcast* dari tetangganya. Sebuah *node* akan mempelajari *node* baru didekatnya dan bagaimana cara menjangkau *node* baru tersebut. Pada suatu saat, setiap *node* akan mengetahui tentang *node-node* lain dan mengetahui bagaimana cara menjangkaunya (Imawan, D. 2009).

Protokol *routing* layaknya sebuah *router* yang berkomunikasi dengan perangkat lain untuk menyebar informasi dan mengijinkan adanya pemilihan rute diantara dua *node* dalam jaringan. pada jaringan *ad hoc* setiap *node* akan memiliki kemampuan layaknya *router* yang meneruskan pesan antar *node* disekitarnya. Untuk itu dibutuhkan protokol *routing* untuk membantu tiap-tiap *node* melakukannya (Imawan, D. 2009).

Protokol *routing* untuk jaringan *ad hoc* tentunya berbeda dengan protokol *routing* yang biasa diimplementasikan pada jaringan kabel. Hal ini disebabkan sifat jaringan *ad hoc* yang dinamis, sehingga memiliki topologi yang berubah-ubah, berbeda dengan jaringan kabel yang cenderung tetap (Imawan, D. 2009). Pada protokol *routing* MANET dapat dibedakan menjadi tiga karakteristik, yaitu (Yanuar, G.C. 2016) :

2.2.1 Protokol Routing Proaktif

Pada protokol *proaktif* ini bekerja dengan cara mendistribusikan *routing table* ke seluruh jaringan, jadi masing-masing *node* mempunyai *routing table* yang lengkap, dalam artian sebuah *node* akan mengetahui semua rute ke *node* lain yang berada dalam jaringan tersebut. Saat melakukan *maintenance* terdapat informasi *routing* melalui *routing table* dan melakukan *up-to-date* secara berkala sesuai dengan perubahan topologi, namun metode *proaktif* ini jika diimplementasikan maka akan

menyebabkan konsumsi *bandwidth* yang besar dikarenakan semua *node* melakukan *broadcast routing table* ke semua *node*.

Beberapa contoh protokol *proaktif* yaitu :

a. B.A.T.M.A.N (*Better Approach to Mobile Ad Hoc Network*)
b. OLSR (*Optimized Link State Routing Protokol*)
c. DSDV (*Destination Sequenced Distance Vector*)
d. HSR (*Hierarchial State Routing Protokol*)
e. WAR (*Witness Aided Routing*)

2.2.2 Protokol Routing Reaktif

Protokol *routing reaktif* melakukan proses pencarian *node* tujuan dengan cara *on demand* yang berarti proses pencarian *route* hanya dilakukan ketika *node* sumber membutuhkan komunikasi dengan *node* tujuan. Jadi *routing table* yang dimiliki oleh sebuah *node* berisi informasi *route node* tujuan saja. Namun pada protokol ini akan membangun koneksi apabila *node* membutuhkan *rute* dalam *mentransmisikan* dan menerima paket data, akan tetapi membutuhkan waktu yang lebih besar daripada *routing* protokol *proaktif*, maka metode ini tidak membutuhkan konsumsi *bandwidth* yang terlalu besar dan meminimalis sumber daya baterai.

a. AODV (*Ad Hoc On Demand Distance Vector*)
b. DYMO (*Dynamic MANET On Demand*)
c. DSR (*Dynamic Source Routing*)
d. FSDSR (*Flow State in the Dynamic Source Routing*)
e. ARAMA (*Ant Routing Algorithm for MANET*)
f. BSR (*Backup Source Routing*)

2.2.3 Protokol Routing Hybrid

Protokol *routing hybrid* adalah metode penggabungan kedua protokol antara *routing proaktif* dan *reaktif*.

a. HWMP (*Hybrid Wireless Mesh Protokol*)

b. ZRP (*Zone Routing Protokol*)

c. HRPLS (*Hybrid Routing* Protokol *for Large Scale MANET*)

Bentuk jalur *routing* ini dapat berubah secara signifikan karena mobilitas *node* dalam jaringan *ad-hoc*, jalan yang dibentuk adalah bukan optimal. Perubahan bentuk dapat dimanfaatkan untuk menurunkan jalur *routing* yang lebih baik, jika kita dapat menghindari *signifikan overhead* (proses penemuan rute tambahan) (Venkatesh C, dkk. 2005).

2.3 DSDV (Destination Sequenced Distance Vector)

DSDV termasuk dalam kategori *table driven routing* protokol dalam jaringan *Mobile Ad Hoc*. DSDV menggunakan metode *routing distance vector* yang dilengkapi dengan adanya *sequence number*. Dengan metode *distance vector*, memungkinkan setiap *node* dalam jaringan untuk dapat bertukar *table routing* melalui *node* tetangganya, namun metode ini dapat mengakibatkan terjadinya *looping* dalam jaringan sehingga digunakanlah suatu *sequence number* tertentu untuk mencegah terjadinya *looping* (Ferdianto, I.A. 2013).

Dalam *protokol routing* DSDV, *sequence number* dihasilkan oleh setiap *node* dalam jaringan yang setiap kali mengirimkan pesan dan terjadinya perubahan dalam jaringan. Hal ini dapat disebabkan karena :

- *Update* secara periodic oleh masing-masing *node* dimana setiap *node* akan mengirimkan pesan secara periodik.
- Jika terdapat *node* yang bergerak sehingga *node* tetangga akan mengirimkan pesan ditandai dengan nilai *sequence number* yang baru.

Dengan metode routing DSDV, setiap *node* memelihara sebuah *table forwarding* dan menyebarkan *table routing* ke *node* tetangga. *Table routing* tersebut memuat informasi sebagai berikut :

- Alamat *node* tujuan.
- Jumlah *hop* yang diperlukan untuk mencapai *node* tujuan.
- *Sequence number* dari informasi yang diterima.
- *Install Time*

Table routing akan diperbarui secara periodic dengan tujuan untuk menyesuaikan jika terjadi perubahan topologi jaringan (ada *node* yang bergerak atau berpindah tempat), dan untuk memelihara konsistensi dari *table routing* yang sudah ada. *Sequence number* yang baru akan dihasilkan oleh setiap *node* jika terjadi pembaruhan *table routing*.

Jika *table routing* sudah diperbaharui maka akan dipilih rute untuk mencapai *node* tujuan dengan kriteria sebagai berikut :

a. *Table routing* dengan nilai *sequence number* yang terbaru akan terpilih. *Sequence number* terbaru ditandai dengan nilai *sequence number* yang lebih besar dari yang sebelumnya.
b. Jika dihasilkan *sequence number* yang sama maka dilihat nilai *metric*nya, dan nilai *metric* yang paling kecil akan dipilih.

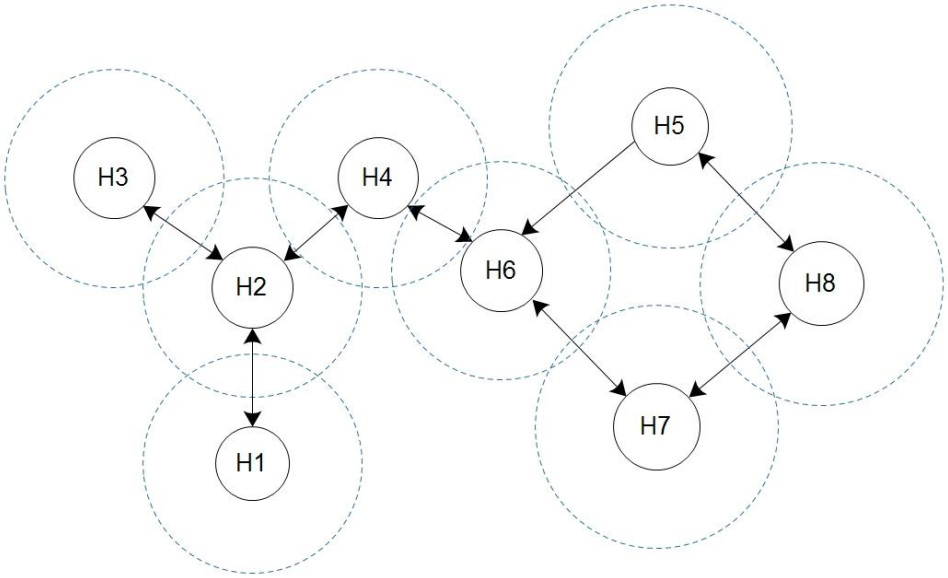
Setiap *node* akan mempunyai sebuah *forwarding table* yang berisi informasi pada *table routing* dan informasi lain seperti *install time*. *Install time* akan berisi interval waktu yang diperlukan untuk mendapatkan *table routing* dari *node* tujuan. Jika *install time* bernilai besar, maka hal tersebut mengindikasikan adanya *link* yang terputus antara *node* asal dan *node* tujuan. *Install time* dijadikan dasar keputusan untuk menghapus rute tertentu yang terputus dengan *node* asal. Dengan penggunaan DSDV maka penghapusan suatu rute tersebut akan jarang sekali dilakukan namun *install time* tetap digunakan untuk memonitoring rute-rute yang terputus dengan *node* asal, dan mengambil langkah yang diperlukan bila hal tersebut terjadi.

Link yang terputus akan ditandai dengan nilai *metric* yang tak terhingga, dan *node* asal akan mengeluarkan *sequence number* ganjil untuk *node* tujuan tersebut. *Sequence number* yang ganjil tersebut akan disebarkan ke *node-node* lain sehingga semua *node* dalam jaringan tersebut mengetahui bahwa ada *link* yang terputus untuk *node* tujuan dengan *sequence number* ganjil tersebut.

Looping dalam jaringan DSDV dapat dihindari dengan penggunaan *sequence number*, dimana setiap *node* untuk setiap perubahan dalam jaringan akan menghasilkan *sequenced number* baru. Jadi *node* lain akan mengetahui

kejadian yang baru terjadi melalui nilai *sequence number*. Semakin besar nilai *sequence number* maka pesan yang diterima semakin baru. *Sequence number* yang lebih kecil menandakan bahwa kejadian tersebut sudah tidak *up to date*.

Gambar 2.2 merupakan contoh jaringan DSDV. Tabel 2.1 merupakan *tabel routing* yang dihasilkan oleh *node* H6. Metode *routing* DSDV memiliki sifat setiap *node* yang berada dalam jaringan akan memelihara sebuah tabel *forwarding* dan menyebarkan tabel *routing* ke *node* tetangganya.

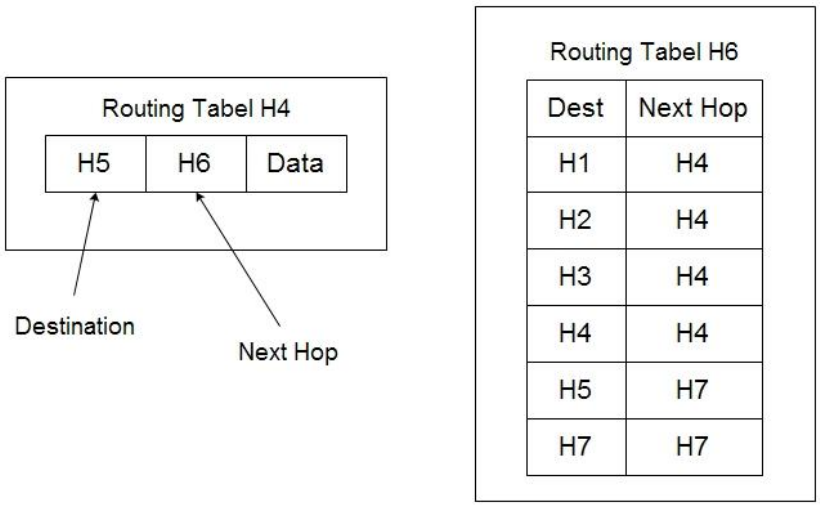


Gambar 2.2 Contoh Jaringan DSDV

Tabel 2.1 Tabel *routing node* H6 (Ferdianto, I.A. 2013).

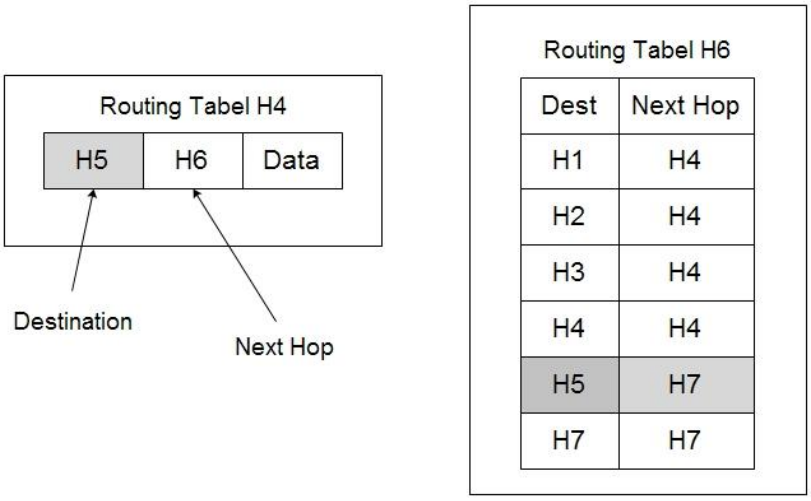
Destination	Next Hop	Metriks	Seq Number	Install Time
H1	H4	3	S406_H1	T001_H6
H2	H4	2	S128_H2	T001_H6
H3	H4	3	S564_H3	T001_H6
H4	H4	1	S710_H4	T002_H6
H5	H7	3	S392_H5	T001_H6
H6	H6	0	S076_H6	T001_H6
H7	H7	1	S128_H7	T002_H6
H8	H7	2	S050_H8	T002_H6

Gambar 2.3 sampai gambar 2.6 menunjukkan prosedur pengiriman paket *routing* pada DSDV. Gambar 2.2 memperlihatkan *node* H4 ingin mengirim paket ke *node* H5. *Node* H4 mengecek *tabel routing* untuk menentukan node H6 yang merupakan *node* berikutnya untuk *routing* paket ke *node* H5. *Node* H4 kemudian mengirim paket ke *node* H6.



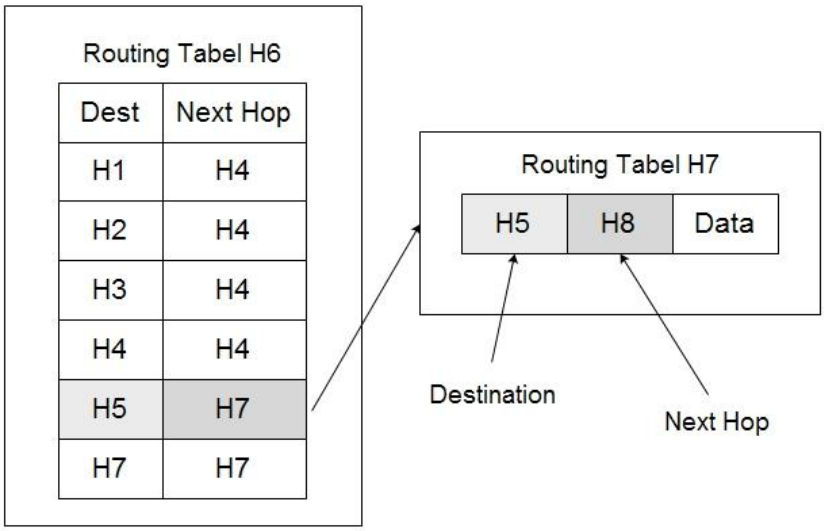
Gambar 2.3 *Node* H4 mengirim paket ke *node* H6 (Ferdianto, I.A. 2013).

Gambar 2.4 memperlihatkan *node* H6 mengecek *tabel routing* yang dimilikinya untuk menentukan node H7 merupakan *node* berikutnya untuk pengiriman paket dari *node* H4 ke *node* H5.



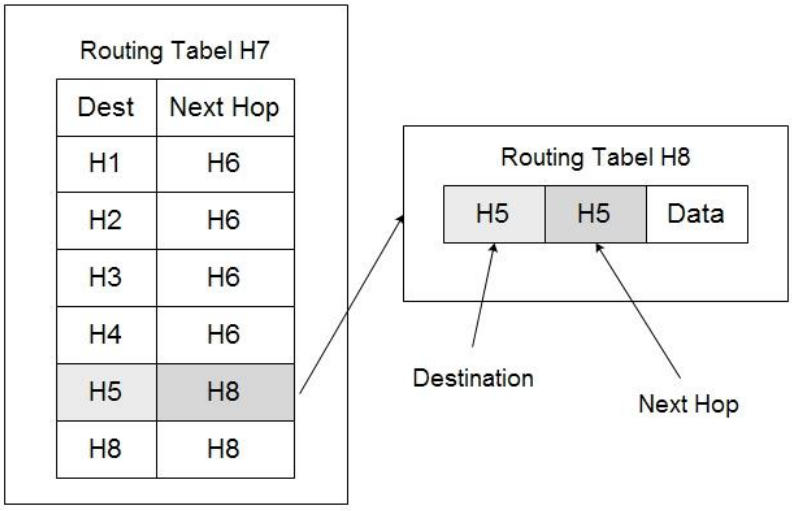
Gambar 2.4 *Node* H6 mengecek *tabel routing*nya (Ferdianto, I.A. 2013).

Gambar 2.5 memperlihatkan *node* H6 meneruskan paket ke *node* H7. Prosedur rute paket tersebut diulang sampai paket dari *node* H4 tiba menuju *node* H5.



Gambar 2.5 *Node* H6 meneruskan paket ke *node* H7 (Ferdianto, I.A. 2013).

Gambar 2.6 menunjukkan *node* H7 meneruskan paket ke *node* H8. *Node* H8 kemudian mengirim paket menuju *node* tujuan yaitu *node* H5.



Gambar 2.6 *Node* H7 meneruskan paket ke *node* H8 (Ferdianto, I.A. 2013).

Gambar 2.6 juga menunjukkan *next hop* dari *node* H8 yaitu *node* tujuan H5. Dengan demikian paket yang dikirimkan dari *node* H4 menuju node H5

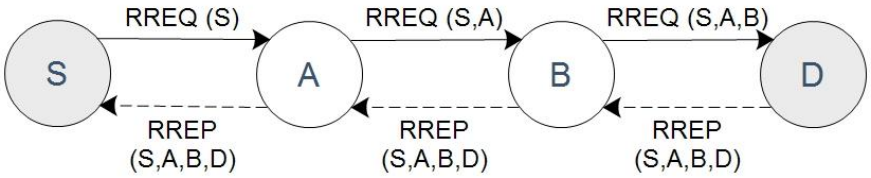
berhasil dengan sebanyak 4 *hop* dengan melewati *node* H6, *node* H7, *node* H8 dan menuju *node* H5 (Ferdianto, I.A. 2013).

2.4 DSR (Dynamic Source Routing)

Dynamic Source Routing (DSR) adalah protokol *routing* yang efisien dan sederhana dirancang khusus untuk digunakan dalam *multi-hop* jaringan *nirkabel ad hoc* mobile. Jaringan sudah mengatur konfigurasi diri sendiri, tidak membutuhkan jaringan infrastruktur atau administrasi. Antara *node* tidak secara langsung dalam transmisi nirkabel berbagai satu sama lain. Semua *routing* ditentukan secara otomatis dan dipelihara oleh *routing* protokol *Dynamic Source Routing* (DSR). Karena jumlah atau urutan antara *hop* yang diperlukan untuk mencapai tujuan dapat berubah setiap saat, topologi jaringan yang dihasilkan cukup banyak dan cepat berubah. Protokol *Dynamic Source Routing* (DSR) memungkinkan *node* secara dinamis menemukan sumber rute di beberapa jaringan *hop* dalam jaringan *ad hoc* (Johnson, dkk. 2001).

Mekanisme protokol DSR dalam melakukan pengiriman paket data dari sumber ke tujuan memerlukan waktu cukup lama melalui proses *route discovery* dan *route maintenance.* Pada *route discovery* terdapat dua kegiatan RREQ dan RREP, sedangkan *route maintenance* mendeteksi adanya perubahan topologi. Jika kemudian terjadi *route eror* (RERR) akan melakukan *route discovery* ulang. Dalam DSR pengaturan rute menurut sumber. Identitas semua *node intermediate* dimasukkan dalam *header packet.* ketika *node* sumber harus mengirimkan paket ke tujuan yang tidak dilihat dalam algoritma penentuan rute yang dipesan, maka ini mengawali proses penemuan rute untuk mencari rute atau beberapa rute dengan menggunakan teknik *broadcast*. Setelah menemukan rute-rute yang diperlukan, sumber akan memulai mentransmisikan paket data dengan menggunakan rute yang ditemukan. *Flooding* merupakan bentuk dari *broadcast* tersebut. Di dalam *flooding,* permintaan rute tetap berlangsung hingga *time to live* (TTL) bidang mencapai nol atau seluruh jaringan terhubung (Al-Rodhaan, dkk. 2010).

Protokol DSR ini terdiri dari dua mekanisme utama, yaitu *Route Discovery* (pencarian rute) dan *Route Maintenance* (pemeliharaan rute) (Wahanani, H.E. 2013) :

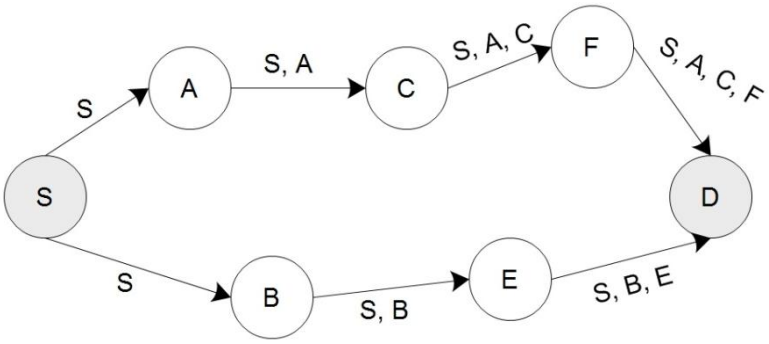


Gambar 2.7 Mekanisme Protokol DSR (Wahanani, H.E. 2013)

2.4.1 Mekanisme Route Discovery

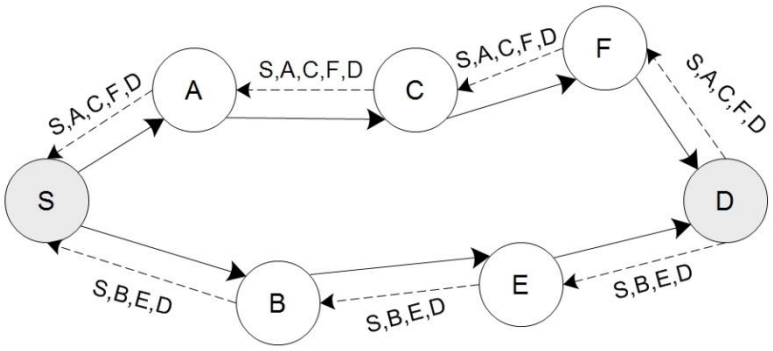
Route discovery adalah suatu mekanisme pada protokol yang berfungsi untuk melakukan pencarian *path* (jalur) secara dinamis dalam jaringan *ad hoc*, baik secara langsung di dalam *range* transmisi ataupun dengan melewati beberapa *node intermediate*. Penentuan *path* ini terbagi menjadi dua bagian yaitu *Route Request* (RREQ) dan *Route Reply* (RREP).

Ketika sebuah *node* sumber (S) ingin mengirim beberapa data ke *node* tujuan (D) dimulai dari pencarian rute. *Route discovery* melakukan *broadcast*, yaitu paket RREQ ke jaringan. *Node intermediate* ketika menerima Paket RREQ melihat ke dalam *cache*-nya untuk mengetahui apakah telah ada beberapa rute ke tujuan, jika ada maka balasan ke pengirim dengan mengirim paket RREP (balasan rute), dimana mengandung rute. Jika *node intermediate* tidak memiliki rute, maka melakukan *broadcast* permintaan ulang setelah menambahkan alamat ke dalam rute sumber (Gambar 2.3). Ketika *query* mencapai *node* tujuan, yang menemukan *node* yang dipesan pada urutan hop dalam paket RREQ dan menggunakan itu untuk memberikan paket RREP ke pengirim (Wahanani, H.E. 2013).



Gambar 2.8 *Route Request* (Wahanani, H.E. 2013)

Pada Gambar 2.8 paket RREQ yang dipancarkan oleh *node* S diterima oleh *node* B. Pada *node* B, paket RREQ yang diterima diperiksa apakah sebelumnya pernah singgah di B. Karena ternyata paket RREQ tersebut belum pernah singgah di B, maka paket RREQ di*forward* dengan cara *flooding* oleh *node* B. Sebelum di*forward*, pada paket RREQ disisipi *identifier* B sehingga pada paket RREQ yang dipancarkan oleh *node* B tersebut terdapat catatan [S,B] yang menunjukkan paket RREQ telah menempuh rute S→B. Hal yang sama dilakukan ketika paket RREQ tersebut singgah ke *node* E. Dengan demikian, ketika paket RREQ tersebut dipancarkan oleh *node* E, pada paket RREQ terdapat catatan yang berisi [S,B,E] yang menunjukkan paket RREQ tersebut telah menempuh rute S→B →E. begitu pula jika dari [S,A,C,F] menunjukkan paket RREQ tersebut telah menempuh rute S→A→C→F. Ketika paket RREQ sampai pada *node* yang dituju yaitu *node destination* D, maka paket tersebut tidak di*forward*. *Node* D membalas paket RREQ tersebut dengan paket RREP yang ditujukan kepada *node* S. Rute yang ditempuh oleh paket RREP merupakan kebalikan rute yang ditempuh paket RREQ. Ingat bahwa rute yang ditempuh paket RREQ dapat diketahui dengan memeriksa catatan *node* yang tersimpan pada paket RREQ tersebut. Pada RREP yang dikirim oleh D juga disisipkan rute dari *node* S menuju *node* D.



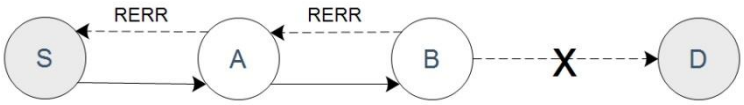
Gambar 2.9 *Route Reply* (Wahanani, H.E. 2013)

Pada Gambar 2.9 ditunjukkan bahwa *node* D menerima paket RREQ [S,A,C,F,D] dan RREQ [S,B,E,D]. Kemudian *node* D membalas paket RREQ tersebut dengan RREP [S,A,C,F,D] dan RREP [S,B,E,D] yang dikirim kepada *node* S. Rute yang ditempuh oleh RREP [S,A,C,F,D] adalah D→F→C→A→S dan rute yang ditempuh oleh RREP [S,B,E,D] adalah D→E→B→S. *Node* S yang menerima RREP [S,A,C,F,D] dan RREP [S,B,E,D] akan mengetahui rute yang harus ditempuh untuk mengirim paket data ke *node* D. *Node* S dapat memilih salah satu rute S→A→C→F→D atau S→B→E→D untuk mengirim paket data. Ketika *node* S mengirim paket-paket data ke *node* D, *node* S menyisipkan informasi rute yang harus ditempuh oleh paket-paket data tersebut untuk menuju *node* tujuan D (Wahanani, H.E. 2013).

2.4.2 Mekanisme Route Maintenance

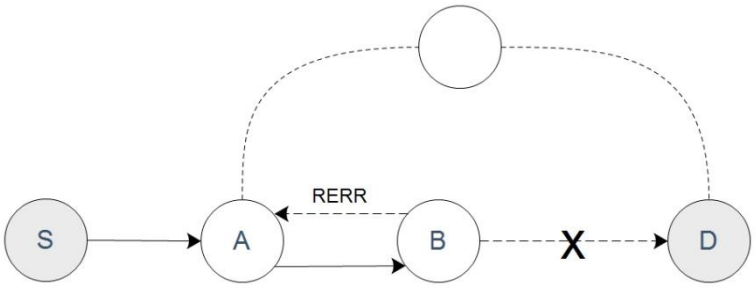
Protokol DSR memiliki mekanisme *route maintenance* dimana sumber mendeteksi adanya perubahan topologi jaringan sehingga pengiriman paket mengalami *kongesti* atau mekanisme yang dipanggil ketika sebuah jalur komunikasi rusak atau gagal terdeteksi selama transmisi data. Hal ini disebabkan karena salah satu *node* yang terdaftar dalam rute sebelumnya bergerak menjauh dari range *node* yang lain. Versi saat ini dari protokol DSR biasa di kenal dengan nama data *salvation* (penyelamatan data). Saat *node* mendeteksi masalah pada rute yang ada, paket RERR akan dikirim pada *node* pengirim. Saat paket

RERR diterima, *hop* ke *node* yang menjauh akan dihilangkan dari *route cache*. Kemudian rute lain yang masih tersimpan di *cache* akan digunakan. Jika tidak ada rute lagi maka protokol DSR akan melakukan proses *route discovery* lagi untuk menemukan rute baru.



Gambar 2.10 Ilustrasi *Route Error* (Wahanani, H.E. 2013)

Pada Gambar 2.10, 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 DSR, 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*.



Gambar 2.11 Penyelamatan data (Wahanani, H.E. 2013)

Gambar 2.11 menjelaskan bahwa ketika *node* bergerak lebih dekat ke *node* tujuan D maka probabilitas mendapatkan rute alternatif yang lebih dan mengurangi waktu untuk memulai *route discovery* dan mendapatkan balasan jika *route discovery* dimulai dari *node* B itu sendiri. Antara B dan S mungkin ada *node intermediate* A yang

memiliki rute ke *node* tujuan D. Jadi, bukannya kesalahan pengiriman rute kembali ke S, B dapat mengirimkan paket kembali ke A dengan paket RERR dalam paket data (diasumsikan bahwa B tidak memiliki jalur alternatif untuk *node* tujuan D dalam *cache*). Dalam cara ini *node* A kemudian akan meneruskan paket saat ini dan selanjutnya ke *node* tujuan D dengan rute alternatif yang telah di *cache*. Skema ini mengurangi *delay* yang akan terjadi (Wahanani, H.E. 2013).

2.5 ZRP (Zone Routing Protokol)

Zone Routing Protokol (ZRP) adalah salah satu dari contoh *hybrid routing* protokol dan pengertian *hybrid routing* protokol sendiri adalah kombinasi dari kedua tipe *routing* protokol yaitu *routing* protokol *proaktif* dan *routing* protokol *reaktif*. ZRP bekerja berdasarkan zona dengan menggunakan konsep zona terbatas menggunakan fitur dari *routing* protokol *proaktif,* sedangkan zona luar menggunakan fitur dari *routing* protokol *reaktif*. ZRP terdiri dari dua *sub routing* protokol utama yaitu *Intra Zone Routing Protokol* (IARP) dan *Inter Zone Routing Protokol* (IERP). IARP mengacu pada jaringan padat yang menjadi batas dari zona *routing* protokol *proaktif* sedangkan IERP mengacu pada jaringan zona luar dari *routing* protokol *reaktif*. IARP mempertahankan informasi topologi jaringan dengan selalu meng*update* jalur ketika *node* berada didalam zona dan IERP hanya bekerja ketika *node* tujuan berada diluar zona (Arinatal, Y.A. 2015).

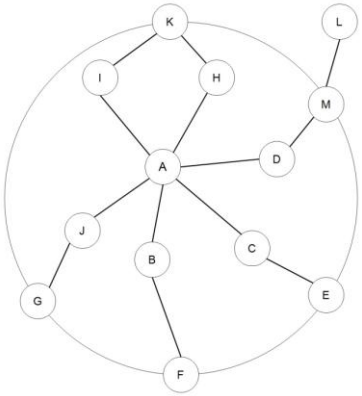
Secara garis besar, konsep ZRP adalah membangun zona di jaringan pada setiap *node* sehingga pada sebuah jaringan memungkinkan banyak sekali zona-zona yang dibangun oleh setiap *node*. Untuk *node* yang berada didalam wilayah geografis yang sudah di tentukan sebelumnya yang selanjutnya akan disebut *radius*, akan dikatakan bahwa *node* tersebut berada dalam zona *routing node* tersebut. Untuk *routing* yang berada didalam zona, digunakan pendekatan *routing proaktif*. Untuk *routing* yang berada diluar zona, digunakan pendekatan *routing reaktif*. Jadi dapat disimpulkan bahwa ZRP menggabungkan beberapa karakteristik protokol *proaktif* dan beberapa

karakteristik protokol *reaktif*, dengan mempertahankan informasi zona *intra* secara *proaktif* dan informasi antar zona secara *reaktif*, menjadi satu untuk mendapatkan solusi yang lebih baik untuk jaringan MANET (Adiwicaksono, S. 2017).

Secara umum, cara kerja ZRP adalah setiap *node* membuat zona pada masing-masing *node*nya. Lalu untuk daerah yang masih didalam zona, maka rute menuju *node* tersebut akan ditemukan, namun ketika *node* tujuan berada di luar zona *routing*, maka akan dilakukan prosedur penemuan rute. Besaran setiap zona tergantung pada *radius* yang didefinsikan pada jumlah *hop* (loncatan) pada setiap *node*nya (Adiwicaksono, S. 2017).

Pada Gambar 2.12, zona *routing* dari dari *node* A dengan radius adalah 2 hop. Node yang termasuk dalam zona *routing* adalah semua *node* kecuali *node* L, karena posisi *node* L yang berada diluar zona *routing node* A. Penentuan zona *routing* tidak ditentukan dari jarak fisik, tetapi dari *hop* (loncatan). Pada zona *routing* terdapat 2 jenis *node* yaitu *peripheral node* dan *interior node* (Adiwicaksono, S. 2017).

Node yang berada pada jarak maksimum radius yang sudah ditentukan disebut *peripheral node*, dan *node* yang berada pada posisi kurang dari radius (berada didalam zona) disebut *interior node*. Pada Gambar 2.12, *Peripheral Node* adalah *node* E, F, G, K, M dan *Interior Node* adalah B, C, D, H, I, J. *Node* L berada diluar zona *node* A (Adiwicaksono, S. 2017).

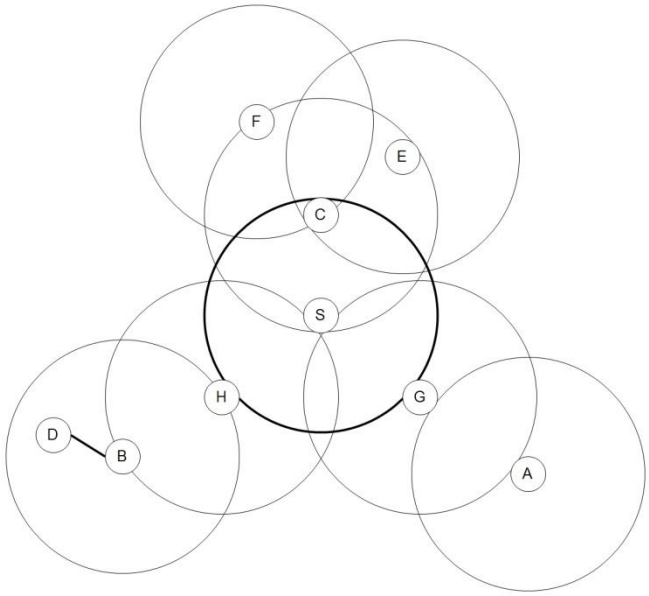


Gambar 2.12 Contoh Zona *Routing Node* A dengan radius 2 *hop* (Adiwicaksono, S. 2017)

Untuk cara kerja pembentukan rute dari ZRP pertama *node* sumber mengirimkan permintaan rute ke *Peripheral Node* miliknya. Permintaan rute berisi alamat sumber, alamat tujuan dan *sequence number*, lalu setiap *peripheral node* akan mengecek zonanya apakah *node* tujuan ada di zonanya. Jika pada zona *peripheral node* tidak terdapat tujuannya, maka *peripheral zone* akan menambahkan alamatnya kepada paket *route-request* dan meneruskan paket kepada *peripheral node*nya. Jika *node* tujuan berada didalam zonanya, maka *peripheral node* akan mengirimkan *route reply* kembali menuju *node* sumber. *Node* sumber menggunakan jalur yang disimpan dalam paket *route-reply* untuk mengirim paket data ke tujuan. Pada ZRP, protokol *routing proaktif* lokal (dalam zona) disebut *IntrA-zone Routing* Protokol (IARP), dan protokol *routing reaktif* global (diluar zona) disebut *IntEr-zone Routing Protokol* (IERP). IARP memelihara informasi *routing* dari *node-node* yang berada dalam zona *routing* sebuah *node*. *Route discovery* dan *route maintenance* dilakukan oleh IERP. Bila diperlukan penemuan global, jika topologi zona lokal diketahui, maka hal tersebut bisa digunakan untuk mengurangi lalu lintas. Untuk mem-*broadcast* paket, ZRP menggunakan konsep *Bordercasting*, yang layanannya disediakan oleh *Bordercasting Resolution Protokol* (BRP) (Adiwicaksono, S. 2017).

Bordercast Resolution Protokol (BRP) merupakan layanan pengiriman paket yang digunakan pada ZRP untuk mengarahkan *route request packet* dari IERP menuju ke *border node*. Proses mengarahkan *route request packet* menuju *border node* disebut *bordercast*. BRP bertanggung jawab untuk meneruskan RRP yang dibentuk IERP ke *border node*. Walaupun penerima dari RRP adalah *border node*, BRP tetap mengirim RRP pada setiap *hop*. BRP akan melacak *node* yang sudah tercakup oleh RRP. Ketika sebuah *node* menerima RRP, *node* tersebut menandai *node* tetangga dari *node* yang melakukan *bordercast* sebagai *node* yang sudah tercakup oleh RRP. Jika *node* penerima adalah *border node* dari *node* yang melakukan *bordercast*, maka *border node* tersebut akan menjadi *node* yang akan melakukan *bordercast* yang baru dan *node* tetangganya akan tercakup oleh RRP. BRP menggunakan

tabel *routing* IARP dari *node* yang melakukan *bordercast*. Ketika BRP menerima RRP baru untuk di-*bordercast*, maka RRP dari *node* pengirim akan mencakup *node* tetangga dari *node* pengirim dan RRP dikirim ke *border node*. BRP bisa memutuskan pengiriman, bila *node* menerima RRP yang sama dari *node* lain (Anthoni, J., dkk. 2014).



Gambar 2.13 Contoh Kerja IERP (Anthoni, J., dkk. 2014)

Sebagai contoh pencarian rute ditunjukkan pada Gambar 2.13. *Node* sumber S ingin mengirimkan paket menuju *node* tujuan D. *Node* S mencari *node* D di dalam zona *routing* milik S. Jika *node* D ada di dalam zona *routing* S, maka *node* S sudah mengetahui rute menuju *node* D. Jika sebaliknya, S melakukan *bordercast* sebuah *route request packet* menuju semua *border node* yang berada dalam zona *routing* S (*node* C, G, dan H). Setiap *node* tersebut tidak menemukan *node* D berada dalam zona *routing* masing-masing dan meneruskan paket tersebut ke *border node* dalam zona *routing* masing-masing. Selanjutnya, *node* H mengirim paket tersebut ke *node* B dengan zona *routing* mencakup *node* D. Lalu *node* D membalas paket tersebut, mengirimkan *route reply* setelah menambahkan alamat *node* D tersebut dan rute dari sumber menuju tujuan dengan rute S-H-B-D (Anthoni, J., dkk. 2014).

2.6 Parameter Pengukuran

Berdasarkan pada parameter pengukuran, untuk mengetahui kinerja dari ketiga protokol *routing* tersebut, maka ditentukan beberapa parameter yang digunakan untuk evaluasi pada penelitian ini yaitu (Windianto, W., dkk. 2015) :

a. *Packet Delivery Ratio* (PDR)

Merupakan perbandingan antara jumlah paket data yang diterima oleh *destination node* dengan jumlah paket data yang dikirim oleh *source node*. PDR dinyatakan dalam persen.

b. *Delay*

Merupakan waktu rata-rata yang dibutuhkan oleh paket data yang dikirimkan oleh *source node* dengan waktu penerimaan paket data oleh *destination node*. *Delay* dinyatakan dalam *millisecond* (ms).

c. *Konsumsi Energi*

Merupakan jumlah energi yang dibutuhkan oleh *node* untuk melakukan proses transmisi data pada jaringan.

d. *Packet Loss*

Merupakan jumlah paket yang tidak berhasil dikirimkan ke tujuan selama transmisi.

e. *Routing Overhead*

Merupakan perbandingan antara total jumlah paket *routing* yang dikirim dengan total jumlah paket data yang diterima. *Routing overhead* dinyatakan dalam persen.

2.7 Penelitian Sebelumnya

Berikut ini merupakan beberapa penelitian sebelumnya tentang *routing* DSR dan DSDV :

1. Penelitian yang dilakukan oleh (Permana, M.Y. dkk, 2010) yang berjudul "*Analisis Pengaruh Penggunaan Protokol Routing AODV, DSDV, dan ZRP Pada Performansi Jaringan Ad Hoc Hibrid".* Pada penelitian ini ZRP memiliki kinerja yang lebih baik dibandingkan

dengan dua *routing* protokol 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* protokol 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* protokol 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.

2. Penelitian yang dilakukan oleh (Anggraini, S.D, dkk. 2017) yang berjudul "*Analisis Perbandingan Performansi Protokol Routing AODV dan DSR Pada Mobile Ad-Hoc Network (MANET)"*. Pada penelitian ini menggunakan simulator *OPNET Modeler 14.5* dengan parameter performansi yang diukur antara lain *latency*, *jitter*, *throughput* dan *packet loss* untuk layanan *realtime* berupa *video conferencing* dan layanan *non-realtime* berupa FTP dengan variasi ukuran data kecil dan data besar. Dari hasil penelitian diperoleh Nilai tertinggi parameter *throughput* yaitu untuk layanan *video conferencing* pada protokol *routing* AODV, besarnya *throughput* yang diterima di setiap *node* pada

protokol *routing* AODV sebesar 17517,56 bit/sec, sedangkan besarnya *throughput* pada protokol *routing* DSR sebesar 372720,67 bit/sec. Hal ini dikarenakan pada protokol *routing* DSR menerapkan *source routing* untuk menentukan rute (*route dicovery*) dan untuk melewatkan paket melalui *hop node* (*hop by hop*). Semakin besar nilai *throughput* maka semakin baik jaringan tersebut bekerja. Nilai tertinggi parameter *jitter* rata-rata yaitu pada protokol *routing* DSR. Nilai parameter *jitter* AODV sebesar 1,40 ms, sedangkan nilai parameter *jitter* pada DSR sebesar 2,73 ms. Hal ini dikarenakan pada protokol *routing* DSR membutuhkan waktu yang lebih lama untuk menemukan rute dan menerapkan *source routing*, seluruh urutan *routing* terletak pada header paket. Semakin kecil nilai *jitter* maka semakin jernih layanan *voice* pada *video conferencing* karena tidak ada variasi *delay*. Nilai *packet loss*, pada protokol *routing* AODV untuk layanan *video conferencing* sebesar 25.503%, lebih baik dibandingkan *routing* DSR sebesar 30,860%. Hal ini disebabkan oleh proses pencarian jalur yang panjang dan lama pada protokol AODV dan juga pengaruh jarak antar *node*. Semakin jauh jarak *node* pengirim dengan *node* penerima, maka paket yang hilang akan semakin besar. Dari seluruh simulasi diperoleh hasil protokol *routing* AODV lebih baik dibandingkan DSR dilihat dari nilai parameter *latency*, *throughput*, dan *jitter*, kecuali parameter *packet loss*. Sedangkan untuk layanan terbaik yang digunakan pada jaringan MANET berupa FTP dengan beban kecil (*low load*).

3. Penelitian yang dilakukan oleh (Amilia, F. 2014) yang berjudul *"Analisis Perbandingan Kinerja Protokol Dynamic Source Routing (DSR) dan Geographic Routing Protokol (GRP) Pada Mobile Ad Hoc Network (MANET)"*. Penelitian ini menggunakan parameter *Throughput, Delay, Load, Media Access Delay, Data Dropped* dan *Network Load.* Berikut ini merupakan hasil dari simulasi : Protokol GRP menghasilkan *throughput* yang lebih besar pada 25 *node* dengan nilai 18.127.650,67 *bit/sec*, sedangkan DSR memiliki nilai *throughput*

5.760.504 *bit/sec*. Hal ini membuktikan bahwa GRP memiliki kemampuan laju pengiriman data lebih baik dibandingkan DSR. Protokol DSR memiliki *delay* yang lebih besar dengan nilai 0,00388 *sec*, dibandingkan dengan protokol GRP yang memiliki nilai 0,00538 *sec*. Dengan ini membuktikan bahwa DSR memiliki kinerja yang buruk dalam proses pencarian rute, sehingga menghasilkan *delay* yang lebih besar. Protokol DSR memiliki nilai *load* lebih besar dengan nilai 121.610.044,44 *bit/sec*, sedangkan GRP memiliki nilai 15.818.457,78 *bit/sec* pada skenario 50 *node*. Dengan demikian protokol DSR lebih baik dibandingkan protokol GRP karena protokol DSR dapat merutingkan paket dari pengirim ke penerima lebih cepat. Protokol GRP memiliki *media access delay* lebih kecil dengan nilai -2,427 *sec* dan DSR dengan nilai 1,691 *sec* pada skenario 50 *node*, Sehingga GRP lebih baik dibandingkan DSR.

4. Penelitian yang dilakukan oleh (Sidharta, Y. 2013) yang berjudul *"Perbandingan Unjuk Kerja Protokol Routing Ad Hoc On-Demand Distance Vector (AODV) dan Dynamic Source Routing (DSR) Pada Jaringan MANET".* Penelitian ini menggunakan parameter *Throughput, Delay, Packet Delivery Ratio (PDR), Jitter*, *Packet Loss* dan *Routing Overhead*. Berikut ini merupakan hasil dari simulasi : Protokol *Routing* AODV menghasilkan nilai *throughput* yang selalu lebih besar yaitu 0,00703 pada 10 *node*, 0,00616 pada 25 *node* dan 0,00611 pada 50 *node* dibandingkan DSR dengan nilai 0,00574 pada 10 *node*, 0,00592 pada 25 *node* dan 0,00612 pada 50 *node*. Dengan ini *routing* AODV memiliki throughput yang lebih baik daripada DSR. Pengaruh penambahan jumlah *node* dan jumlah koneksi tidak terlalu signifikan pada *routing* protokol DSR untuk parameter jaringan *delay, jitter*, dan *routing overhead*. Sedangkan penambahan jumlah *node* dan jumlah koneksi sangat berpengaruh terhadap kinerja *routing* protokol AODV untuk semua parameter jaringan yang diukur. Pada scenario

penambahan 50 *node*, kinerja routing AODV dan DSR untuk parameter *packet delivery ratio* (PDR) dan *packet loss* hampir sama.

5. Penelitian yang dilakukan oleh (Imawan, D. 2009) yang berjudul *"Analisis Kinerja Pola-Pola Trafik Pada Beberapa Protokol Routing Dalam Jaringan MANET".* Pada penelitian ini melibatkan *routing* protokol AODV, DSR, dan DSDV dengan parameter *routing overhead, packet delivery ratio (PDR)* dan *average delay*. Hasil dari simulasi ini menunjukkan bahwa protokol DSR ini sangat cocok diimplementasikan pada jaringan yang besar dan tingkat mobilitas tinggi, karena DSR memiliki kelebihan, yaitu :

- DSR memiliki performa yang lebih baik daripada AODV dan DSDV terhadap perubahan kapasitas jaringan yaitu ditunjukkan dengan nilai *routing overhead* yang relatif kecil.
- DSR memiliki nilai *PDR* yang relatif tinggi yaitu di atas 80% baik oleh perubahan kapasitas jaringan maupun pada beragam tingkat mobilitas jaringan. Kekurangan dari DSR yaitu *average delay* meningkat sangat besar pada peningkatan volume trafik.

Protokol AODV memiliki kelebihan dalam hal :

- Nilai *PDR* relatif tinggi baik oleh perubahan kapasitas jaringan, tingkat mobilitas, maupun tingkat volume trafik jaringan, yaitu berkisar di atas 82%.
- *Average delay* kecil pada beberapa tingkat volume trafik.

Sedangkan kekurangan dari AODV, yaitu :

- *Routing overhead* meningkat cukup tajam seiring meningkatnya kapasitas jaringan dan mencatat nilai tertinggi dibandingkan dengan DSR dan DSDV.
- kurang cocok diterapkan pada kondisi jaringan dengan mobilitas tinggi, karena *routing overheadnya* meningkat pada mobilitas tinggi.

Routing overhead pada protokol DSDV hanya dipengauhi oleh kapasitas jaringan, sehingga perubahan mobilitas dan volume trafik, *routing overhead* DSDV cenderung konstan. Kelebihan lain dari DSDV yaitu

memiliki *average delay* yang kecil pada segala kondisi kapasitas jaringan dan tingkat volume trafik. Nilai *PDR* DSDV lebih rendah dibandingkan dengan *PDR* AODV dan DSR, dan mengalami penurunan cukup signifikan pada jaringan dengan mobilitas tinggi.

2.8 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* protokol baik pada jaringan kabel maupun *wireless* (secara lokal maupun dengan satelit) (Fahriani, N., dkk. 2012).

2.8.1 Perkembangan Awal

Network Simulator (NS) dibangun sebagai varian dari REAL Network Simulator pada tahun 1989 di UCB (University of California Berkeley). Dari awal tim ini dibangun sebuah perangkat lunak simulasi jaringan *internet* untuk kepentingan riset interaksi antar protokol dalam konteks pengembangan protokol *internet* pada saat ini dan masa yang akan datang.

2.8.2 Kelebihan NS 2

Kelebihan dari NS-2 yaitu sebagai perangkat lunak simulasi pembantuanalisis 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 scenario 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.

2.8.3 Simulasi Yang Menggunakan NS2

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.

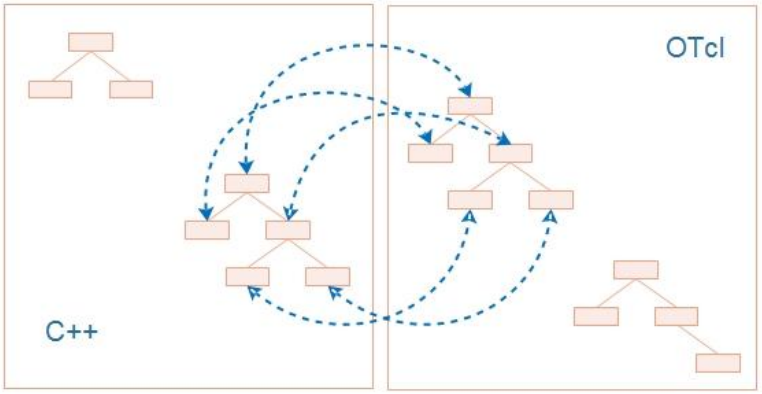
2.8.4 Konsep Dasar NS2

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 *wireless*.

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.

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 2.14 berikut ini.



Gambar 2.14 Hubungan C++ dan Otcl

Keterangan :

- *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*

Bahasa C++ digunakan pada *library* karena C++ mampu mendukung *runtime* simulasi yang cepat, meskipun simulasi melibatkan jumlah paket dan sumber data dalam jumlah besar.

Bahasa Tcl memberikan respon *runtime* yang lebih lambat daripada C++, namun jika terdapat kesalahan *syntax* dan perubahan *script* berlangsung dengan cepat dan interaktif. *User* dapat mengetahui letak kesalahannya yang dijelaskan pada *console*, sehingga *user* dapat memperbaiki dengan cepat. Karena alasan itulah bahasa ini dipilih untuk digunakan pada *script* simulasi.

2.8.5 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*.

2.8.6 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.

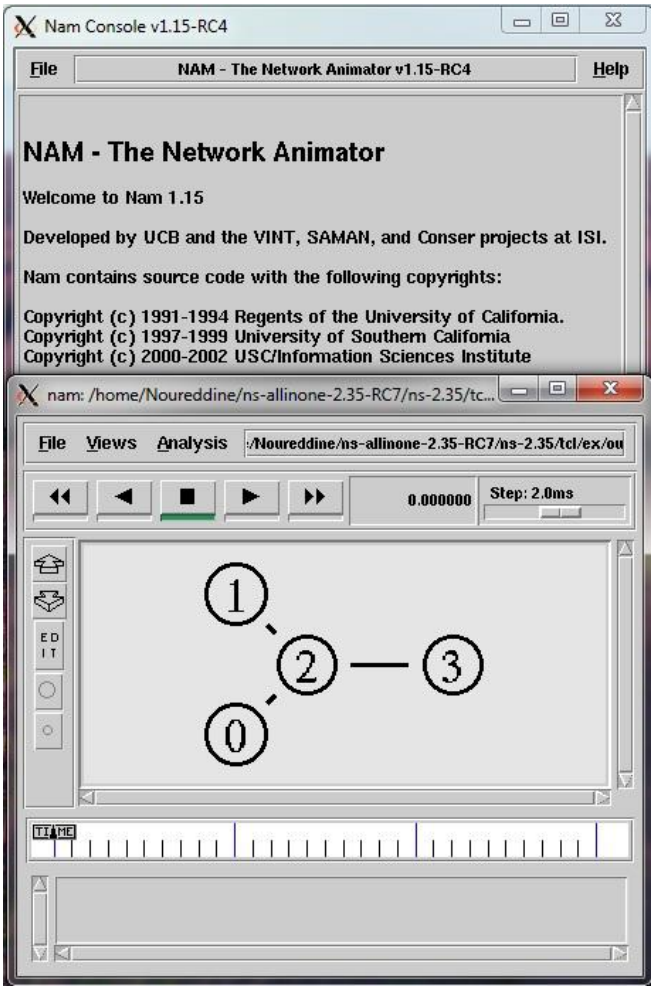
Contoh : [root@accessnet your_folder]#ns simulasi.tcl

2.8.7 Output Simulasi NS2

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 2.15.



Gambar 2.15 *NAM Console*