BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis

Rute merupakan salah satu kunci penting dalam jaringan MANET karena sifatnya yang sangat dinamis. Mobilitas *host* bisa menyebabkan sering terjadinya perubahan topologi yang tidak bisa diprediksi. Proses penemuan rute sering kali membanjiri jaringan dengan paket permintaan rute untuk mencari rute di seluruh jaringan. Hal ini dapat mengakibatkan *delay* yang tinggi untuk mencapai jalur. Parameter yang mempengaruhi kinerja protokol penentuan rute jaringan *ad hoc* adalah beberapa *node* di dalam area, perilaku *node* yang mengubah konektivitas (bergerak secara bebas sebagai kelompok, frekuensi perubahan topologi), dan kualitas *link*. Rancangan strategi penentuan rute yang efisien dan *reliabel* memang merupakan masalah yang sangat menantang karena sumber-sumber yang sangat terbatas di dalam MANET (Fahriani N, Djanali S, Shiddiqi AM, 2012).

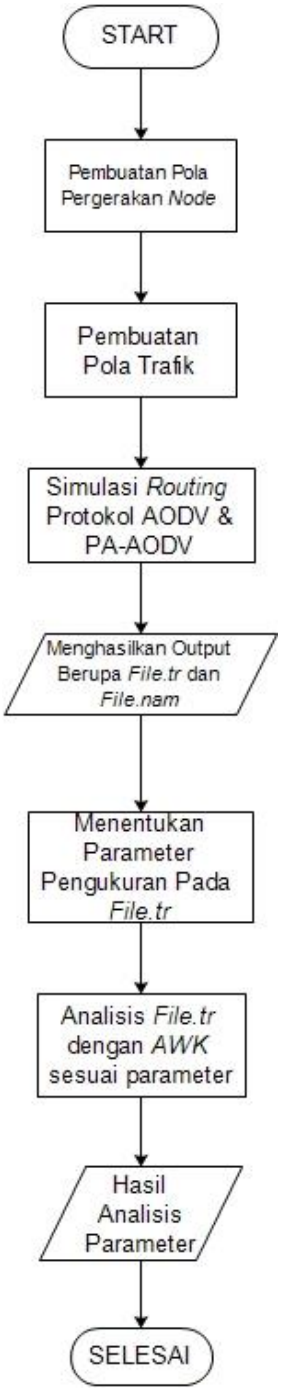
Permasalahan *routing* protokol AODV pada proses pencarian rute melalui proses yang cukup lama yaitu pada saat *node* sumber mengirimkan paket *route request* (RREQ) apabila terjadi kegagalan *link* antar *node* maka *node* sumber akan melakukan proses pengiriman paket RREQ ulang.

3.2 Hasil Analisis

Dengan melihat kinerja *routing* AODV yang dalam pembentukan rute membutuhkan waktu yang cukup lama adalah dengan cara efisiensi pencarian rute diantara *node* yang tidak membebani *link*. Oleh sebab itu untuk mendukung informasi optimasi *link-link* yang menyusun rute digunakan algoritma *Path Aware Short* dengan memastikan bahwa *link* yang akan dilalui dalam kondisi baik dan pencarian rute yang paling optimum dengan parameter waktu tempuh yang paling minimal sehingga tidak perlu melakukan proses pengiriman paket *route request* (RREQ) ulang.

Hasil proses efisiensi akan dilakukan dengan menggunakan *tools network simulator 2* dimana tingkat performansi akan dibandingkan nilai parameter

average End-to-end delay, normalized routing load, dan *packet delivery ratio (PDR)*antara *routing* AODV dengan *routing* AODV yang sudah dimodifikasi dengan algoritma *path aware short*. Pada gambar 3.1 akan dijelaskan diagram alir analisis *routing*.



Gambar 3.1 Diagram Alir Analisis *Routing*

Penjelasan gambar 3.1:

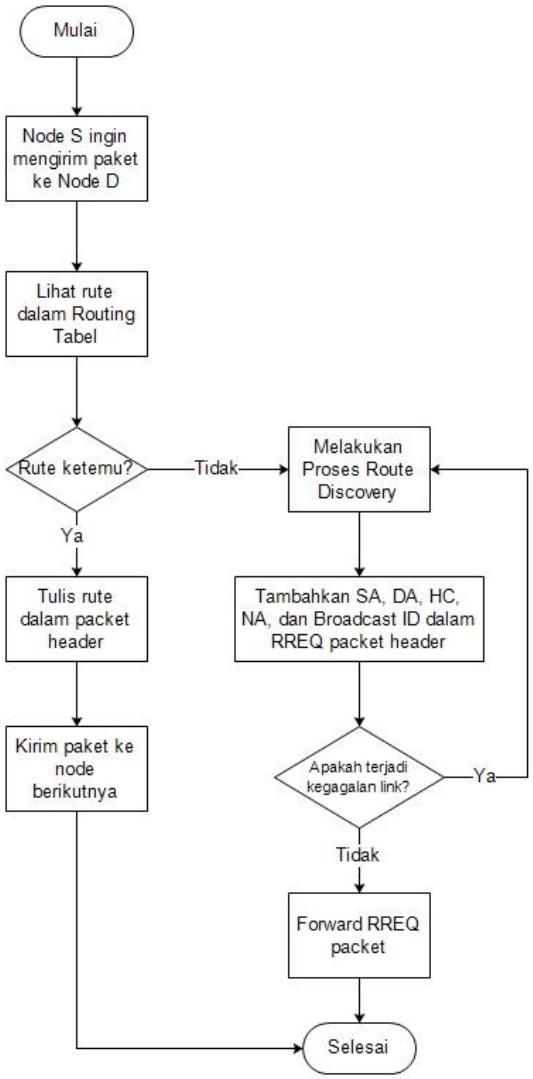
1. Langkah pertama dibuat pola pergerakan *node*, pada langkah ini *node-node* bergerak secara acak dan dinamis.
2. Pembuatan pola trafik dalam menentukan jumlah *node*, jumlah keseluruhan paket data yang ditransmisikan, juga menentukan *node* mana yang akan menjadi *node* sumber dan *node* mana yang akan menjadi *node* tujuan.
3. Proses simulasi *routing* AODV dan juga algoritma *path aware* AODV untuk didapatkan output berupa *file.tr* dan *file.nam*. Dimana *file.tr* merupakan *file trace* yang digunakan analisa numerik dan *file.nam* merupakan tampilan grafis simulasi.
4. Membuat perhitungan parameter *packet delivery ratio*, *delay* dan *normalized routing load*.
5. Menganalisis parameter dengan menggunakan *file.awk* terhadap *file.tr* yang akan menghasilkan sebuah angka dari nilai-nilai parameter dari performansi *routing* AODV dan algoritma *path aware* AODV.

3.3 Algoritma Routing Protokol

Pada penilitian ini ada 2 algoritma yang akan di analisis diantaranya algoritma *routing* AODV dengan algoritma *path aware short* yang dimasukkan kedalam *routing* AODV. Berikut dibawah ini 2 algoritma *routing* protokol:

3.3.1 Algoritma Routing 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 EH, 2014).

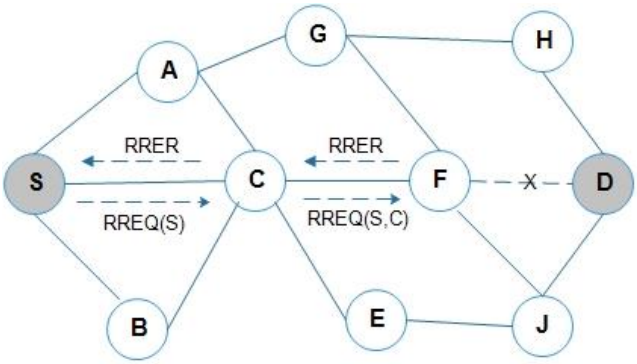


Gambar 3.2 *Flowchart routing* AODV

Berikut penjelasan gambar 3.2:

1. Dimulai ketika *node* S (*source*) ingin mengirimkan paket ke *node* D (*Destination*).
2. Lihat dalam *routing cache* apakah ada rute menuju *node* D apa tidak.
3. Jika ada maka *node* S akan menuliskan rute kedalam *packet header* dan paket akan langsung dikirimkan ke *node* D.
4. Jika tidak ada rute menuju *node* D maka akan melakukan proses *route discovery* dan *node* S akan membroadcast paket *route request* (RREQ) ke *node* tetangga.

5. Pada RREQ paket *header* ditambahkan SA (*source addres*), DA (*destination addres*), HC (*hop count*), NA (*neighbor addres*), dan Broadcast ID.
6. Apakah terjadi kegagalan *link*, jika ada sebuah *link* antar *node* terputus maka akan melakukan proses *route discovery* ulang. Mengulangi langkah nomer 4.
7. Jika tidak ada kegagalan *link* antar *node* maka paket RREQ diteruskan sampai ke *node* D.



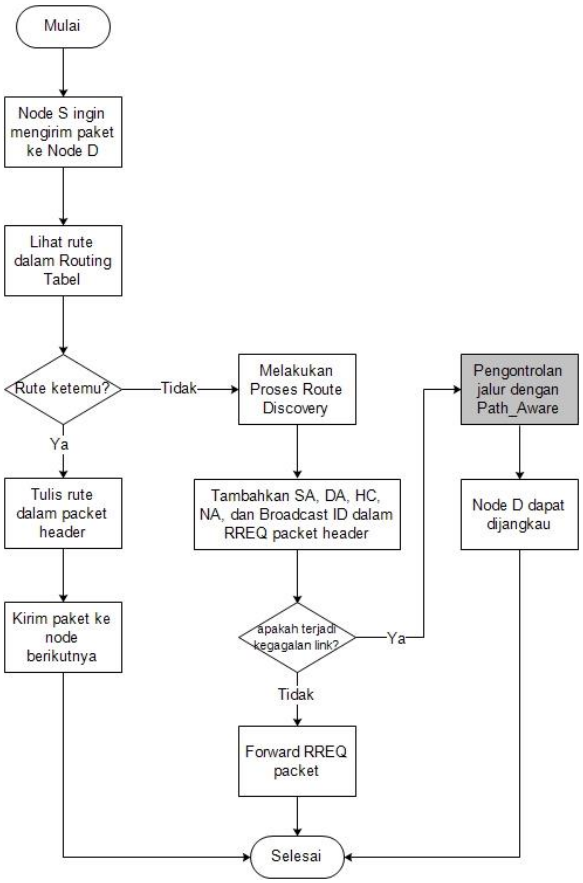
Gambar 3.3 Contoh AODV

Pada gambar 3.3 dijelaskan ketika ada suatu paket yang akan dikirim, maka *node* S akan melihat apakah ada rute menuju *node* tujuan D pada route chache. Ketika *node* S sudah ada rute menuju node tujuan maka akan mengirimkan *route request* menuju *node* D, ketika dalam pengiriman terjadi kegagalan *link* seperti pada *node* F yang kehilangan rute menuju ke*node* D maka akan dikirimkan pesan *route error* kembali menuju ke*node* S.

3.3.2 AODV dengan metode PATH-AWARE

Modifikasi yang dilakukan pada protokol AODV adalah dengan melakukan perubahan pada proses *route discovery* dan *route maintenance*. Dengan dilakukan efisiensi adanya kegagalan *link* di dalam pembentukan rute dengan mengontrol *delay*. Pada gambar 3.3 algoritma AODV yang

telah dimodifikasi dengan menggunakan Algoritma Path Aware Short akan dinamakan dengan PA-AODV.



Gambar 3.4 *Flowchart* PA-AODV

Penjelasan gambar 3.3:

1. Algoritma PA-AODV dimulai dari *node* S (*source)* yang akan mengirim paket ke *node* D (*destination*).
2. *Route table* akan dicek terlebih dahulu apakah rute sudah ada atau belum.
3. Jika rute sudah ada maka *node* S akan menuliskan rute tujuan kedalam *packet header*, dan akan langsung dikirim ke *node* D.
4. Jika *node* S belum memiliki rute menuju *node* D maka akan melakukan proses *route discovery*.
5. *Node* S akan melakukan *broadcast* sebuah paket *route request* (RREQ) ke *node* tetangganya.

6. Pada RREQ paket *header* ditambahkan SA (*source addres*), DA (*destination addres*), HC (*hop count*), NA (*neighbor addres*), dan Broadcast ID.
7. Pada saat pengiriman RREQ apakah terjadi kegagalan *link*, jika tidak ada kegagalan link maka paket akan diteruskan sampai ke *node* D.
8. Apabila terjadi kegagalan *link* maka*node* S akan memonitor rute berikutnya dengan Algoritma *Path-Aware*, yaitu mekanisme pengontrolan jalur sehingga *node* berikutnya dapat terjangkau dan bisa untuk dilewati. Dimana *node* disekitarnya akan mencari rute alternatif untuk meneruskan paket yang dikirim dengan *delay time* terkecil. *Path-Aware* adalah teknik umum yang bekerja dengan pengaturan rute dasar. Untuk optimalisasi pengguna rute akan dilihat pada *normalized routing load* yang paling kecil diantara beberapa jalur. Untuk mendukung algoritma *Path-Aware* yang diajukan, setiap paket membawa bidang *hop-count* (HC) dalam headernya. HC field diawali dengan nol di *node* sumber dan ditingkatkan satu di setiap *hop* yang diambil paket. Untuk setiap paket, alamat tujuan (DA), alamat sumber (SA), dan HC dapat diperoleh dari packet header. Informasi ini disimpan sebagai suatu kesatuan ,emurut kesatuan perbandingan *hop*. Format setiap entry dalam kesatuan adalah <SA, DA, HC, NA>, dimana NA adalah alamat tetangga tempat paket disiarkan. Sebelum SA mengirim paket ke *node hop* pertama, entry berikut informasi dimasukkan dalam kesatuan perbandingan *hop* SA: <SA, DA, 0, SA>.

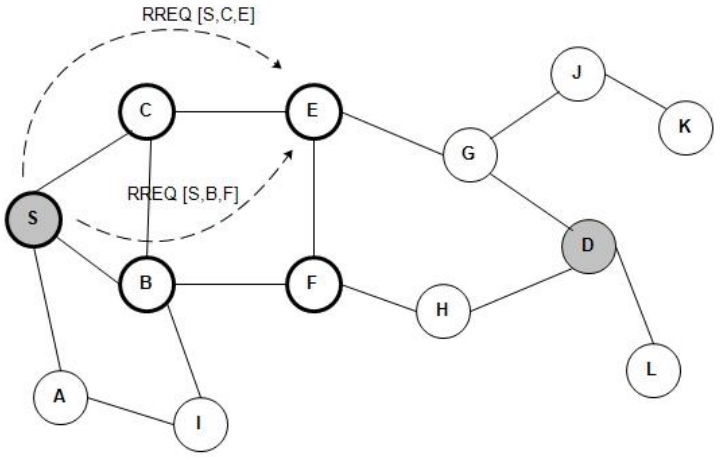
Dapat dilihat pada gambar 3.5. Algoritma *Path Aware-Short.*

```
Saat node i menerima paket p maka
START
 a. Jika node i adalah DA (destination addres), paket dikonsumsi
 b. Bandingkan SA (source address) dan DA dari semua entry dalam
    kesatuan perbandingan hop
 c. Jika tidak ada kesesuaian entry, simpan SA, DA, HC (hop count),
    dan NA (neighbor address) dalam kesatuan perbandingan hop
 d. Jika paket ditujukan ke i sebagai node hop berikutnya, proses
    paket untuk forwarding.
 e. Jika sesuai dengan entry (SA, DA, HC, NA) berarti node i
    melakukan:
    • Mengirim pesan ke NA untuk mengupdate tabel pengaturan rute
      alamat hop berikutnya untuk node tujuan (DA)
END
```

Gambar 3.4 Algoritma *Path Aware-Short* (Fahriani N, Djanali S, Shiddiqi AM, 2012).

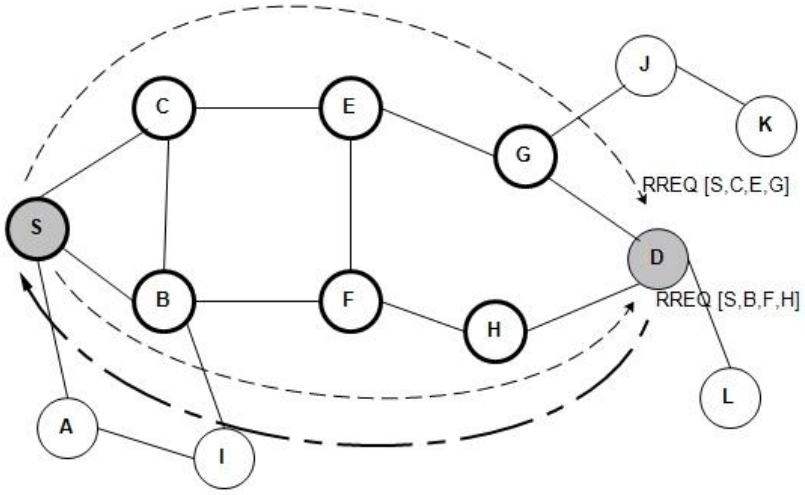
Berikut proses *route discovery* untuk mendapatkan efisiensi rute (Fahriani N, Djanali S, Shiddiqi AM, dkk, 2012). :

1. Sebuah *node* S (*source*) akan mengirim paket ke *node* D (*destination*) pada sebuah MANET. Bila S tidak mengetahui rute menuju D, S akan melakukan *route discovery*. Proses *route discovery* dilakukandengan melakukan *flooding* paket RREQ oleh S.
2. *Node* intermediate i (selain *node* S dan D) yang menerima RREQ akan
   - Memeriksa *node* yang telah dilalui paket RREQ. Jika RREQ pernah singgah di i, maka RREQ akan di drop karena terjadi pergeseran.
   - Sebelum memforward RREQ, *node* i menyisipkan identifiernya pada RREQ yang menandakan RREQ tersebut pernah singgah di i.



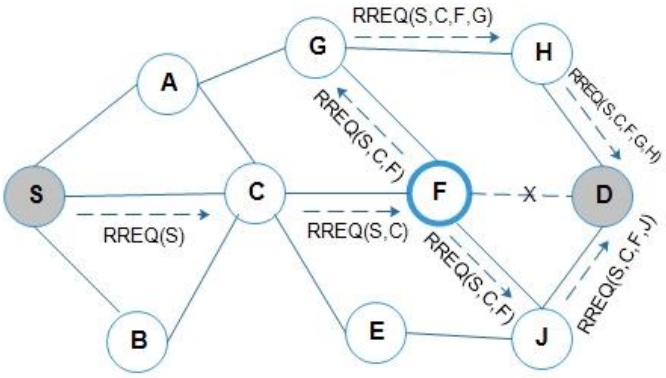
Gambar 3.6 ilustrasi aturan forwarding oleh *node intermediate*

3. *Nodedestination* D yang menerima RREQ akan membalas dengan mengirim *route reply* (RREP).



Gambar 3.7 ilustrasi aturan pengiriman RREP oleh *node* destination

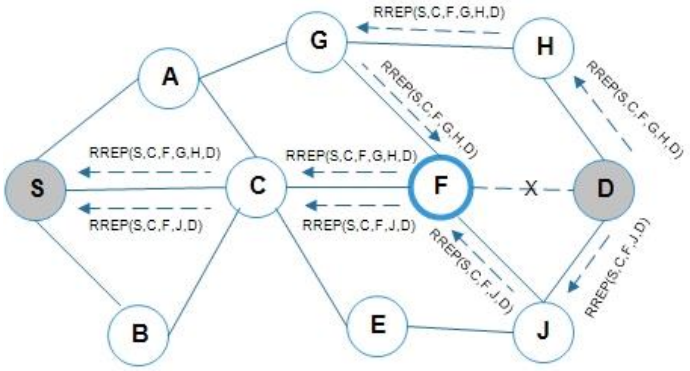
4. *Node source* S yang menerima beberapa RREP akan memilih rute yang efisien.



Gambar 3.8 RREQ AODV dengan Path Aware

Pada gambar 3.8 dijelaskan pada saat *node* S mengirimkan RREQ dan sudah mengetahui rute yang dituju yaitu S→C→F→D, pada saat pengiriman RREQ *node* F kehilangan rute menuju *node* D, *node* F tidak mengirimkan pesan RRER kembali ke node S tapi dengan adanya *path aware* dimana *node* F akan menjadi *node intermediate* yang akan melakukan RREQ ulang tanpa harus kembali ke*node* sumber dan mencari

rute menuju *node* tujuan dengan cara membroadcast pesan RREQ ke *node* tetangga.



Gambar 3.9 RREP AODV dengan Path Aware

Pada gambar 3.9 dijelaskan *node* D telah menerima paket RREQ [S,C,F,J,D] dan RREQ [S,C,F,G,H,D]. kemudian *node* D membalas paket RREQ tersebut dengan RREP [S,C,F,J,D] dan RREP [S,C,F,G,H,D] yang dikirim ke *node* S untuk mengetahui rute mana yang harus dituju dengan melihat jangkauan terdekat antar *hop*. Dengan adanya tambahan algoritma *path aware* ini kemungkinan untuk kegagalan *link* (RRER) akan semakin kecil dikarenakan *path aware* akan mencari *node* terdekat sehingga proses pembentukan rute untuk sampai ketujuan bisa terhubung.

3.4 Perancangan Parameter Simulasi

Dalam melakukan proses simulasi perlu digunakan parameter-parameter untuk menghasilkan hasil sesuai dengan kebutuhan.Pada tabel 3.1 di tunjukkan parameter dan skenario yang digunakan untuk simulasi dengan menggunakan Network Simulator 2.30.

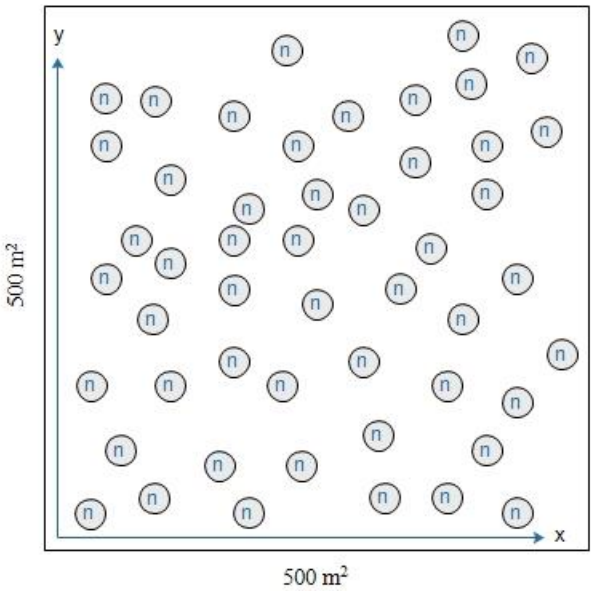
Tabel 3.1 Parameter dan Skenario

Parameter dan Skenario	
Tipe parameter	Nilai Parameter
Packet rate	1 Packet/s
Transmisi range	250 m

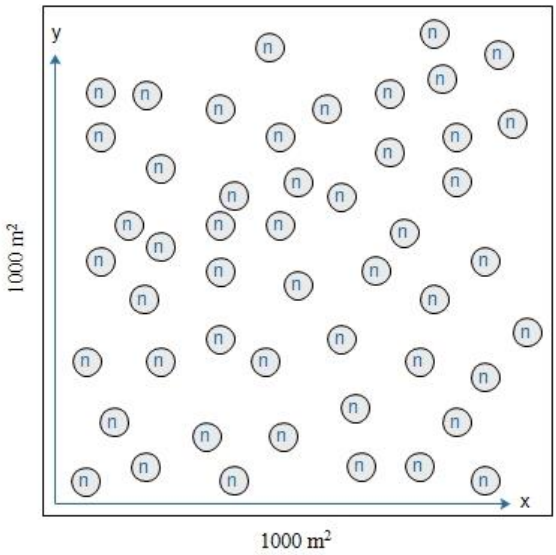
Mac layer	802.11
Jumlah *node*	50 *node*, 100 *node*
Network area	500 x500 m^2, 1000 x 1000 m^2, 1000 x 1500 m^2
Waktu simulasi	5.0 s
Mobility model	Random Waypoint
Propagation	Two Ray Ground
Maximum speed	10 m/s
Maximum connection	5 *node*, 10 *node*, 15 *node*
Pola trafik	CBR
Packet size	512 byte

3.5 Desain topologi simulasi

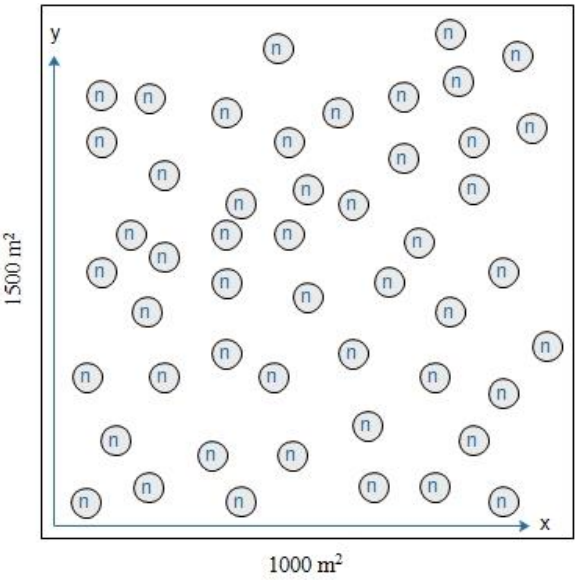
Desain topologi simulasi terbagi menjadi beberapa bagian yaitu topologi *node* 50 dengan *network area* 500 x500 m^2, 1000 x 1000 m^2, 1000 x 1500 m^2. Dapat ditunjukkan pada gambar 3.10, gambar 3.11 dan gambar 3.12.



Gambar 3.10 Topologi 50 *node* dengan *network area* 500 x500 m^2



Gambar 3.11 Topologi 50 *node* dengan *network area* 1000 x1000 m^2

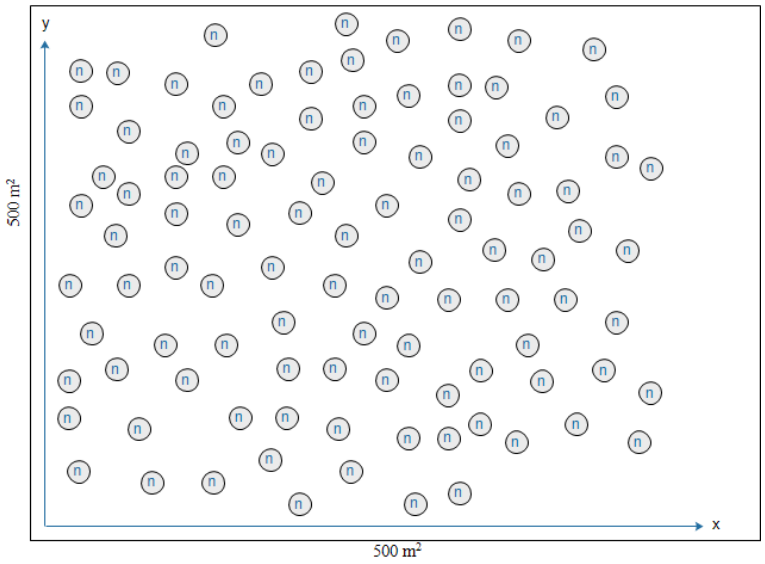


Gambar 3.12 Topologi 50 *node* dengan *network area* 1000 x1500 m^2

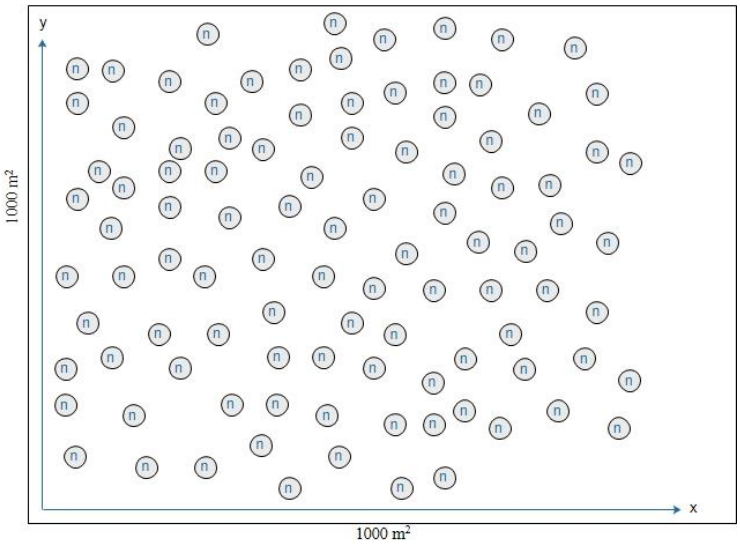
Pada gambar 3.10, gambar 3.11 dan gambar 3.12 dijelaskan bahwa pada topologi *node* 50 yang dimana *node* tersebut diacak secara random dengan

network area yang berbeda-beda yaitu pada gambar 3.10 topologi 50 *node* dengan *network area*500 x500 m^2, pada gambar 3.11 topologi 50 *node* dengan *network area*1000 x1000 m^2 dan pada gambar 3.12 topologi 50 *node* dengan *network area*1000 x1500 m^2.

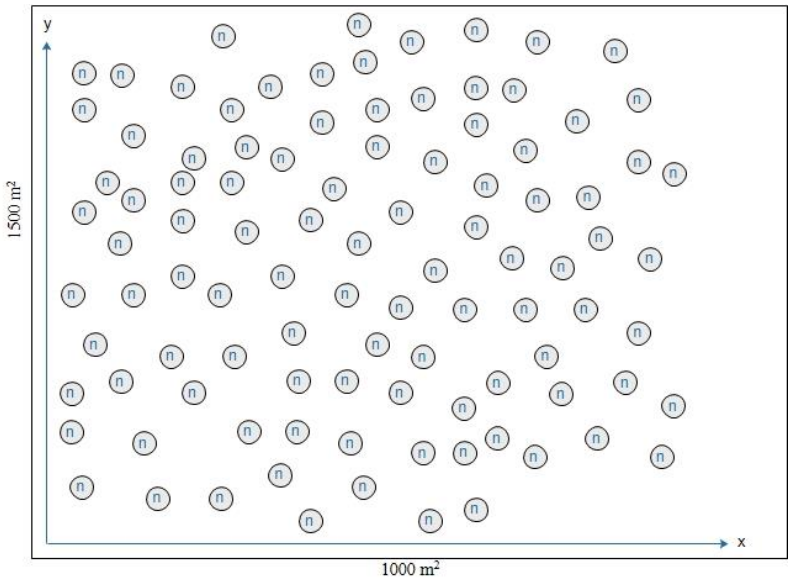
Untuk topologi *node* 100 dengan *network area*500 x500 m^2, 1000 x 1000 m^2, 1000 x 1500 m^2dapat ditunjukkan pada gambar 3.13, gambar 3.14 dan gambar 3.15.



Gambar 3.13 Topologi 100 *node* dengan *network area* 500 x500 m^2



Gambar 3.14 Topologi 100 *node* dengan *network area* 1000 x1000 m^2



Gambar 3.15 Topologi 100 *node* dengan *network area* 1000 x1500 m^2

Pada gambar 3.13, gambar 3.14 dan gambar 3.15 dijelaskan bahwa pada topologi *node* 100 yang dimana *node* tersebut diacak secara random dengan *network area* yang berbeda-beda yaitu pada gambar 3.13 topologi 100 *node* dengan *network area*500 x500 m^2, pada gambar 3.14 topologi 100 *node* dengan *network area*1000 x1000 m^2 dan pada gambar 3.15 topologi 100 *node* dengan *network area*1000 x1500 m^2.

3.6 Skenario Simulasi

Simulasi dilakukan menggunakan scenario jaringan *wireless* standard IEEE 802.11 dengan jangkauan 250 m. variasi pola trafik yang akan digunakan dalam mengukur performa *routing* protokol dalam penilitian ini, antara lain:

1. Simulasi pada penilitian ini menggunakan kapasitas jaringan *network area*500 x500 m^2, 1000 x 1000 m^2, 1000 x 1500 m^2 dengan masing-masing 50 *node* dan 100 *node*.

2. Simulasi pada penilitian ini menggunakan pola pergerakan *node random waypoint* dengan *maximum speed* 10m/s. Paket ini terdiri file *setdest.h* dan *setdest.cc*. dengan penjelasan parameter:
   -n : menyatakan jumlah *node* dalam jaringan
   -p : menunjukkan jeda waktu sebuah *node* dalam keadaan tidak bergerak
   -M : maksimum kecepatan *node* dalam bergerak
   -t : durasi waktu simulasi
   -x : ukuran (dalam sumbu x) topologi jaringan
   -y : ukuran (dalam sumbu y) topologi jaringan
3. Pola trafik yang digunakan dalam penilitian ini adalah *Constan Bit Rate* (CBR). CBR berarti setiap sumber trafik yang berada pada suatu jaringan akan mengirimkan datanya secara terus menerus. Dalam pola trafik ada beberapa parameter yang harus dimasukkan. Diantaranya:
   - Tipe koneksi yang digunakan
   - Jumlah *node* dalam simulasi
   - Jumlah *seed*
   - Maksimum koneksi yang dikehendaki
   - Paket *rate*

3.7 Parameter Pengukuran

Berikut adalah beberapa parameter yang digunakan dalam menguji kinerja PA-AODV terhadap AODV yaitu (Fahriani, N., dkk, 2012):

1. *Packet Delivery Ratio*

   Packet delivery ratio (PDR) adalah rasio antara banyaknya paket yang diterima oleh tujuan dengan banyaknya paket yang dikirim oleh sumber. Berikut rumus untuk menghitung PDR:

   $$Packet\ Delivery\ Ratio\ (\%) = \frac{\sum \text{paket data yang diterima}}{\sum \text{paket data yang dikirim}} \text{ x } 100\%$$

2. *Average end-to-end delay*

Average end-to-end delay adalah jeda waktu antara paket pertama dikirim dengan paket tersebut diterima. Berikut rumus untuk menghitung *Average end-to-end delay* :

$$Average\ Delay(m/s) = \frac{\text{total delay}}{\text{total paket yang diterima}}$$

Dimana

$$delay = \text{Received time} - \text{Sent time}$$

3. *Energy*

   Energy adalah energy yang digunakan oleh *node* untuk berinteraksi dan mengirimkan paket ke semua *node*.

3.8 Analisis Parameter dengan AWK

Proses analisis dengan AWK untuk mengetahui nilai parameter yang sudah ditentukan yaitu *Packet Delivery Ratio* (PDR), *Average End-to-End Delay* dan *energy*. Kemunculan nilai dari parameter tersebut didapatkan dari *script* awk. Instruksi-instruksi dari *awk* terdiri atas *pattern* dan *action* atau gabungan keduanya.

3.9 Spesifikasi Kebutuhan Pembuatan Sistem

Dalam melakukan simulasi dan analisis dibutuhkan spesifikasi perangkat keras dan perangkat lunak.

a. Kebutuhan perangkat keras

   Perangkat keras adalah komponen fisik peralatan yang membentuk sistem komputer, serta peralatan lainyang mendukung komputer dalam menjalankan tugasnya. Adapun rekomendasiperangkat keras yang dibutuhkan untuk menjalankan aplikasi ini adalah :

   1. Prosesor Intel Core i3-3217U 1.8 Ghz
   2. Random Access Memory 2 GB
   3. Monitor VGA atau SVGA 14 inch
   4. Harddisk 500 GB
   5. Keyboard

6. Mouse

b. Kebutuhan perangkat lunak

Sedangkan untuk *spesifikasi software* (kebutuhan perangkat lunak) untuk merancang aplikasi ini adalah:

1. Sistem Operasi Windows 7 Service Pack 1
2. Microsoft Office
3. Edraw Max 7.9
4. Cygwin version 2.738
5. NS-allinone 2.30
6. Network Animator versi 1.15
7. Notepad++