BAB III
METODOLOGI

3.1. Analisis Masalah

MANET merupakan sebuah teknologi yang dapat menghubungkan dua buah *node* atau lebih menggunakan jaringan *wireless* tanpa infrastruktur. Teknologi jaringan MANET diciptakan sebagai antisipasi jika infrastruktur jaringan sedang mengalami gangguan. Dengan jaringan MANET sistem komunikasi yang dilakukan tidak membutuhkan infrastruktur jaringan karena tiap *node* pada jaringan tersebut bersifat *mobile*.

Seperti yang terjadi di Komunitas Trail Adventure Balikpapan, komunitas tersebut mempunyai anggota sekitar kurang lebih 20 orang yang tiap minggunya melakukan perjalanan menjelajahi hutan di sekitar Kota Balikpapan. Ketika melakukan perjalanan tersebut, sangat sering sekali dua hingga tiga pengendara terpisah oleh pengendara lainnya. Komunikasi antar anggota sangat diperlukan ketika hal tersebut terjadi, agar anggota yang terpisah dapat bergabung kembali hingga garis *finish*. Permasalahan muncul ketika komunitas ini memasuki wilayah-wilayah tertentu dimana infrastruktur-infrastruktur seperti penyedia jaringan telepon seluler belum mengembangkan layanannya hingga ke daerah-daerah tersebut. Sehingga proses komunikasi sangat tidak mungkin dilakukan, yang berakibat kegiatan *adventure* komunitas tersebut menjadi tidak menarik. Pada tugas akhir ini, penulis ingin membuat simulasi jaringan terlebih dahulu yang dirancang sesuai dengan parameter-parameter yang ada agar menyerupai keadaan sebenarnya, kemudian menerapkan hasil simulasi ke penerapan jaringan MANET dengan 2 *node*.

3.2. Studi Literatur

Teknologi yang berpotensi dapat menyelesaikan permasalahan sesuai dengan studi kasus Komunitas Trail Adventure Balikpapan adalah MANET. Berdasarkan penelitian sebelumnya yang dilakukan oleh Ahmad (2014) yang berjudul Penerapan Teknologi Jaringan MANET untuk File Transfer, penelitian

tersebut menggunakan MANET sebagai teknologi untuk melakukan *file transfer* antar empat *node* dengan tiga skenario. Pada skenario pertama, jumlah *node* yang digunakan dua *node*, 3 *node* pada skenario kedua, dan 4 *node* pada skenario ketiga. Kesimpulan pengujian jaringan pada skenario satu adalah File Transfer pada infrastruktur jaringan manet dengan dua node dapat bekerja dengan baik tanpa adanya gangguan yang terjadi. Hal ini tidak lepas dari faktor tidak terdapat node gateway pada pengujian skenario ini karena node A dapat langsung terhubung dengan node B. Pada skenario kedua, kesimpulannya adalah transfer file yang terjadi pada node client-server dapat bekerja namun dengan catatan bahwa kecepatan transfer file menurun jika dibandingkan dengan skenario pengujian manet 2 node. Hal ini dikarenakan terdapat node gateway C yang berfungsi sebagai penghubung node A dan node B. Namun tentu saja disini jika node C tidak ada maka komunikasi antara node A dan B tidak akan terjadi sama sekali karena tidak saling menjangkau. Pada skenario ketiga, kesimpulannya adalah sama seperti kondisi skenario pengujian manet 3 node bahwa transfer file antara node client dan node server dapat bekerja namun dengan adanya penurunan kecepatan transfer file. Kecepatan transfer pada skenario ini lebih rendah dibandingkan pada skenario pengujian manet 3 node, hal ini menunjukkan bahwa pada kondisi multihop atau dalam arti lain mempunyai node gateway lebih dari satu akan berimbas pada kecepatan transfer file.

Penggunaan OLSR sebagai protokol *routing* MANET didasarkan pada penelitian sebelumnya yang telah dilakukan oleh Wahyu (2011) yang berjudul Perbandingan Kinerja Protokol AODV Dengan OLSR Pada MANET. Penelitian tersebut bertujuan untuk membandingkan protokol *routing* AODV dengan OLSR pada MANET melalui simulasi dengan menggunakan OPNET. Skenario yang dibuat saat simulasi terdiri dari empat skenario, yaitu:

1. Skenario AODV dengan jumlah *node* 50
2. Skenario OLSR dengan jumlah *node* 50
3. Skenario AODV dengan jumlah *node* 100
4. Skenario OLSR dengan jumlah *node* 100

Parameter yang digunakan untuk mengetahui kualitas jaringan MANET adalah *Delay, Load, Throughput, Packet Delivery Ratio*. Kesimpulan berdasarkan hasil simulasi tersebut antara lain:

1. Hasil simulasi skenario 50 dan 100 *node* menunjukkan bahwa OLSR memiliki kinerja terbaik. Parameter kedua protokol yang mempengaruhi kinerja yaitu pesan *hello*. OLSR memiliki pengaturan pesan *hello* sebesar 2 detik sedangkan AODV sebesar 1 detik.
2. Berdasarkan hasil simulasi skenario 50 *node*, AODV memiliki *delay* sebesar 0,335 detik dan OLSR sebesar 0,079 detik. Pada simulasi skenario 100 *node,* AODV menghasilkan *delay* sebesar 1,186 detik dan OLSR sebesar 0,334 detik, sehingga OLSR lebih cepat dalam mencari rute. OLSR menggunakan jarak terpendek berdasar informasi tabel routing, sedangkan AODV menambahkan jarak (hop) dari 1 *node* ke *node* lain.
3. Berdasarkan hasil simulasi skenario 50 *node*, AODV menghasilkan *load* sebesar 184.416.384 bit dan OLSR sebesar 191.866.476 bit. Pada simulasi skenario 100 *node,* AODV menghasilkan *load* sebesar 368.578.038 bit dan OLSR sebesar 330.149.010 bit. Hal ini menunjukkan AODV skenario 50 *node* lebih sedikit mengirim paket ke jaringan, sehingga AODV lebih cocok untuk jaringan skala kecil. Sebaliknya untuk skenario 100 *node*.
4. Berdasarkan hasil simulasi skenario 50 *node*, AODV memiliki *throughput* sebesar 841.199,38 bps dan OLSR sebesar 1.266.713,82 bps. Pada simulasi skenario 100 *node,* AODV memiliki *throughput* sebesar 068.832,52 bps dan OLSR sebesar 6.158.824,13 bps. Keberhasilan pengiriman paket lebih besar OLSR, karena informasi topologi OLSR selalu diperbarui jika ada perubahan terutama saat *node* bermobilitas. Setiap *node* dalam OLSR dapat menerima informasi topologi lebih dari sekali, sehingga menimbulkan pengulangan pesan *(overhead)*.
5. Berdasarkan hasil simulasi skenario 50 *node*, AODV menghasilkan PDR sebesar 96,02 % dan OLSR sebesar 97,80 %. Pada simulasi skenario 100 *node*, AODV menghasilkan PDR sebesar 76,40 % dan OLSR sebesar 95,67 %. OLSR lebih handal dalam mengirimkan paket jaringan terutama untuk jaringan dengan

jumlah *node* lebih besar, karena OLSR menggunakan jarak terpendek dan tabel *routing* ketika merutekan paket, sedangkan AODV berdasarkan jarak *hop* antar tetangga.

Pemilihan NS-3 sebagai simulator jaringan adalah berdasarkan penelitian sebelumnya yang dilakukan oleh Irawan dan Roestam (2011) yang berjudul Simulasi Model Jaringan Mobile Ad-Hoc (MANET) Dengan menggunakan NS-3. Penelitian tersebut adalah Mensimulasikan jaringan MANET akan memberikan efisiensi dan manfaat yang sangat besar, terutama dalam hal investasi perangkat yang membentuk jaringan semacam ini. Terbentuknya model simulasi MANET akan memungkinkan perancangan dan evaluasi terhadap jaringan bisa dilakukan tanpa harus secara fisik membangun infrastrukturnya terlebih dahulu. NS-3 sebagai sebuah aplikasi *network simulator* memiliki kemampuan untuk memodelkan jaringan yang menerapkan berbagai protokol dan bentuk topologi dengan berbagai skenario. Hal ini memungkinkan untuk dimodelkan karena, selain berbasis sistem operasi terbuka (*open source*), NS-3 juga didukung oleh bahasa C++ untuk menfasilitasi pengembangan model mulai dari bentuk topologi, *node* sampai kepada detail mekanisme protokolnya. Penelitian tersebut menggunakan AODV sebagai protokol *routing*. Perancangan topologi yang dibuat untuk MANET adalah bersifat dinamis, artinya pada MANET *node* juga berfungsi sebagai *router* yang meneruskan paket ke *node* lainnya, sehingga jaringan ini terus berganti (berubah-ubah). Penulis membuat salah satu skenario, yaitu mendefinisikan sebuah *node* yang diam berjumlah satu buah, dan *node* yang bergerak berjumlah 5 buah. Simulasi akan berjalan dengan keadaan semua *node*, baik yang diam atau yang bergerak, akan berbaris dengan panjang barisan maksimal 10 *node* dan akan membentuk baris kedua jika *node* lebih dari 10 dan seterusnya. Jarak antar *node* dalam barisan adalah 1 meter. Kesimpulan hasil simulasi adalah

1. Jaringan MANET bisa disimulasikan menggunakan NS-3 terutama karena dukungan C++ yang memberikan kemudahan untuk menspesifikasikan protokol jaringan MANET dengan lebih detil.
2. Presentasi hasil test dalam bentuk grafis akan memudahkan evaluasi

terhadap suatu desain MANET, sehingga rancangan jaringan bisa dioptimalkan sebelum direalisasikan secara fisik.

3.3. Skenario Simulasi

Simulasi jaringan MANET pada tugas akhir ini menggunakan simulator NS-3. secara umum skenario simulasi yang dirancang merupakan pemanfaatan fungsi jaringan adhoc yang tidak membutuhkan instalasi infrastruktur yang tetap sehingga skenario saat tidak tersedianya infrastruktur bisa diterapkan.

Pada skenario ini komunikasi berjalan dua arah, yaitu dari banyak node (source) ke banyak node (sink) lainnya. Inisialisasi node (source dan sink) dilakukan secara acak.

Alasan penggunaan parameter-parameter seperti pada tabel 3.2 adalah agar ketika simulasi dijalankan, hasil yang didapat sesuai dengan keadaan aslinya. Parameter secara lengkap yang dipakai dalam merancang skenario antara lain (Tabel 3.2):

Tabel 3.1 Parameter Simulasi

Parameter	Protokol OLSR
Simulator	NS-3 versi 3.24.1
Tipe MAC	IEEE 802.11
Data *Rate*	802.11 b (11 Mbps)
Protokol *Transport*	UDP
Jumlah *Node*	3
Jumlah *sink*	2
Waktu Simulasi	60 detik
Transmit Power	13.7 dBm
Besar Paket	140 Byte
Kecepatan	65536 bps / 8 KBps

Instalasi Parameter pada NS-3:

1. Instalasi Data *Rate*:

   Untuk menentukan data *rate* yang digunakan saat simulasi, pertama-tama buat variabel *rate*, kemudian panggil variabel tersebut pada kelas *DataRate.*

```
std::string rate ("65536bps");
Config::SetDefault
("ns3::OnOffApplication::DataRate",       StringValue
(rate));
```

2. Instalasi protokol UDP:

   Untuk menggunakan UDP sebagai protokol transport saat simulasi, perlu dieksekusi perintah berikut, dan tentukan waktu mulai dan berakhirnya.

```
onOffHelper  onoff1  ("ns3::UdpSocketFactory",Address
());
onoff1.SetAttribute       ("OnTime",       StringValue
("ns3::ConstantRandomVariable[Constant=1.0]"));
onoff1.SetAttribute      ("OffTime",      StringValue
("ns3::ConstantRandomVariable[Constant=0.0]"));
```

3. Membuat *node:*

   Pembuatan *node* pada ns-3 adalah menggunakan kelas *NodeContainer* kemudian gunakan fungsi *Create* dalam kelas tersebut.

```
int nWifis = 3;
NodeContainer adhocNodes;
adhocNodes.Create (nWifis);
wifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
YansWifiPhyHelper wifiPhy =
YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss
("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());
```

```
// Add a non-QoS upper mac, and disable rate control
NqosWifiMacHelper wifiMac =
NqosWifiMacHelper::Default ();

wifi.SetRemoteStationManager
("ns3::ConstantRateWifiManager",
                                "DataMode",StringValue
(phyMode),

"ControlMode",StringValue (phyMode));

wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));

wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer   adhocDevices   =   wifi.Install
(wifiPhy, wifiMac, adhocNodes);
```

4. Menentukan jumlah *sink:*

   Untuk menentukan jumlah *sink* adalah dengan memakai kelas *ApplicationContainer*, kemudian atur waktu mulai dan berhenti setiap *node* untuk bertukar data.

```
int nSinks = 2;
for (int i = 0; i <= nSinks; i++)
   for(int a=0;a<=nSinks;a++){
  {
    Ptr<Socket>     sink     =     SetupPacketReceive
(adhocInterfaces.GetAddress (i), adhocNodes.Get (i));

    AddressValue   remoteAddress   (InetSocketAddress
(adhocInterfaces.GetAddress (i), port));
    onoff1.SetAttribute ("Remote", remoteAddress);

    Ptr<UniformRandomVariable>          var          =
CreateObject<UniformRandomVariable> ();
    ApplicationContainer   temp   =   onoff1.Install
(adhocNodes.Get (nSinks-a));
  temp.Start (Seconds (var->GetValue (5.0,5.5)));
  temp.Stop (Seconds (var->GetValue (59.5,60.0)));}}
```

5. Inisialisasi posisi *node*:

   Mengatur posisi awal *node* saat simulasi adalah dengan menggunakan kelas *posistionAlloc* kemudia menambahkan koordinat pada setiap *node*.

```
Ptr<ListPositionAllocator>      positionAlloc      =
CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (500.0, 2460.0, 0.0));
positionAlloc->Add (Vector (800.0, 2060.0, 0.0));
positionAlloc->Add (Vector (1100.0, 1660.0, 0.0));
```

6. Menentukan waktu simulasi:

   Menentukan kapan simulasi akan berakhir dapat diatur menggunakan memanggil fungsi stop pada kelas *ApplicationContainer* kemudian tentukan nilainya dalam satuan detik.

```
double TotalTime = 60.0;
ApplicationContainer    temp    =    onoff1.Install
(adhocNodes.Get (nSinks-a));
    temp.Start (Seconds (var->GetValue (5.0,5.5)));
    temp.Stop (Seconds (var->GetValue (59.5,60.0)));
```

7. Menentukan daya pancar *wireless* (*Tx Power*):

   Menentukan daya pancar *wireless* setiap *node* dapat ditentukan dengan memasukkan variabel txp (nilai daya pancar) ke parameter fungsi *wifiPhy.Set*.

```
double txp = 13.7;
experiment.Run     (nSinks,     txp,     CSVfileName);
}
wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
```

8. Menentukan besar paket yang akan dikirim:

   Besar paket yang akan dikirim adalah 140 *Byte*, yaitu dengan memasukkan nilai tersebut ke parameter fungsi *PacketSize*

```
Config::SetDefault
("ns3::OnOffApplication::PacketSize",StringValue
("140"));
```

9. Mengatur IP *address*:

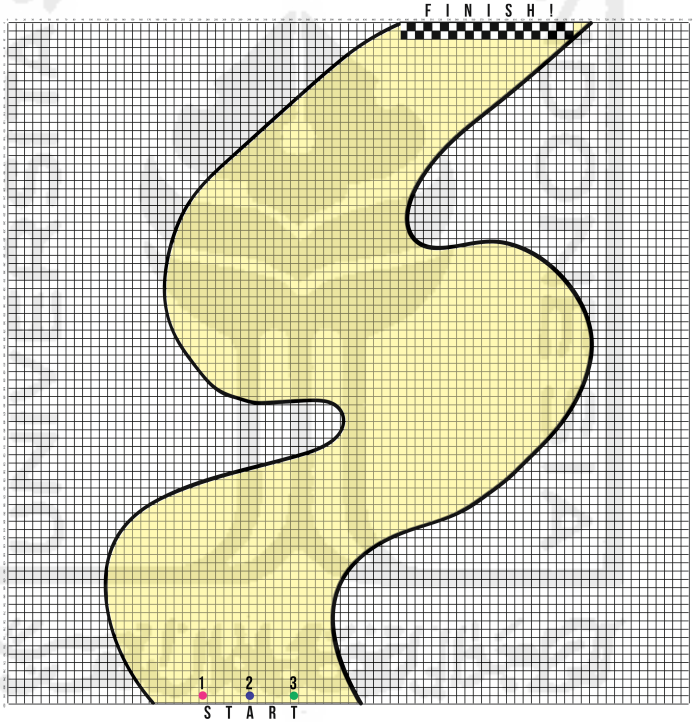
   Mengatur IP *address* tiap *node* adalah dengan menentukan *Network* IP beserta *netmask* nya. Kemudian implementasikan dikelas

Ipv4InterfaceContainer.

```
NS_LOG_INFO ("assigning ip address");
Ipv4AddressHelper addressAdhoc;
addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign (adhocDevices);
```

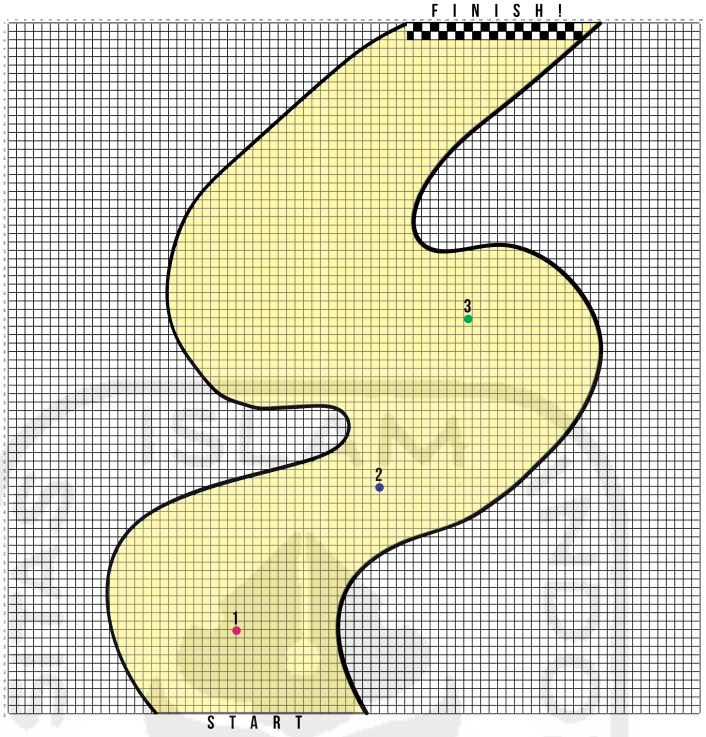
3.3.1. Topologi Jaringan

MANET memiliki karakteristik jaringan yang tidak tetap dan *mobile*. Skenario dengan topologi dan posisi *node* dibagi menjadi dua skenario. Posisi *node* pada skenario satu ditunjukkan oleh Gambar 3.1.



Gambar 3.1 Posisi *node* pada skenario pertama

Pada skenario kedua, kedua *node* akan ditempatkan sesuai dengan posisi pada gambar 3.2.



Gambar 3.2 Posisi *node* pada skenario kedua

Node-node tersebut akan menggunakan IP *address* sebagai berikut:

Tabel 3.2 Daftar IP *address* tiap *node*

Node	IP address
Node 1	10.1.1.1
Node 2	10.1.1.2
Node 3	10.1.1.3

3.3.2. Trace Output

a. *Output Trace File Format .csv*

File dengan format .csv merupakan file yang berisi kumpulan data yang terjadi selama simulasi berlangsung berupa tabel (gambar 3.2). Kumpulan-kumpulan data tersebut berisi parameter seperti:

- *Receive Rate,* merupakan jumlah banyaknya paket yang diterima per detik dikali dengan ukuran paket.
- *Packet Receive,* merupakan berapa kali total sebuah node menerima

paket dari node lain.

SimulationSecond	ReceiveRate(KBps)	PacketsReceived	NumberOfSinks	RoutingProtocol	TransmissionPower(dBm)
0	0	0	3	OLSR	7.5
1	0	0	3	OLSR	7.5
2	0	0	3	OLSR	7.5
3	3.968	62	3	OLSR	7.5
4	6.144	96	3	OLSR	7.5
5	14.144	221	3	OLSR	7.5
6	16.384	256	3	OLSR	7.5
7	21.824	341	3	OLSR	7.5
8	22.72	355	3	OLSR	7.5
9	21.312	333	3	OLSR	7.5
10	20.8	325	3	OLSR	7.5
11	15.232	238	3	OLSR	7.5
12	5.056	79	3	OLSR	7.5
13	7.168	112	3	OLSR	7.5
14	6.272	98	3	OLSR	7.5
15	3.584	56	3	OLSR	7.5
16	2.688	42	3	OLSR	7.5
17	2.752	43	3	OLSR	7.5
18	0.896	14	3	OLSR	7.5
19	1.728	27	3	OLSR	7.5
20	2.112	33	3	OLSR	7.5

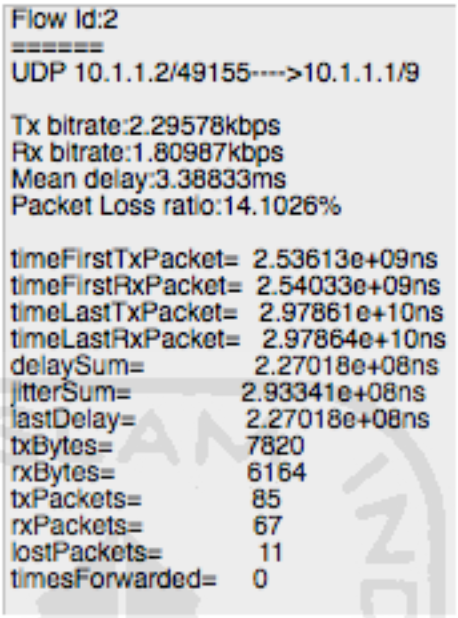
Gambar 3.3 Isi file dengan format .csv

Untuk menghasilkan *file trace* dengan format .csv, perlu dieksekusi perintah di bawah ini pada *script* file utama:

```
std::ofstream out ("nama_file.csv".c_str(),
std::ios::app);
```

b. *Output Trace File Format .flowmon*

File output dengan format *.flowmon* merupakan *file* yang berisi data informasi mengenai parameter yang terjadi antara hubungan satu *node* ke *node* lainnya selama simulasi berlangsung. Parameter-parameter yang ada dalam *file .flowmon* antara lain seperti jenis protokol yang digunakan, IP address sumber dan tujuan, *transmit bitrate, receive bitrate,* rata-rata *delay*, *packet loss ratio,* jumlah *jitter,* dan jumlah *delay*. File dengan format *.flowmon* dapat di eksekusi dengan menggunakan aplikasi NetAnim. Gambar 3.3 merupakan hasil eksekusi *file .flowmon*.



Gambar 3.4 Hasil eksekusi *file* dengan format .flowmon

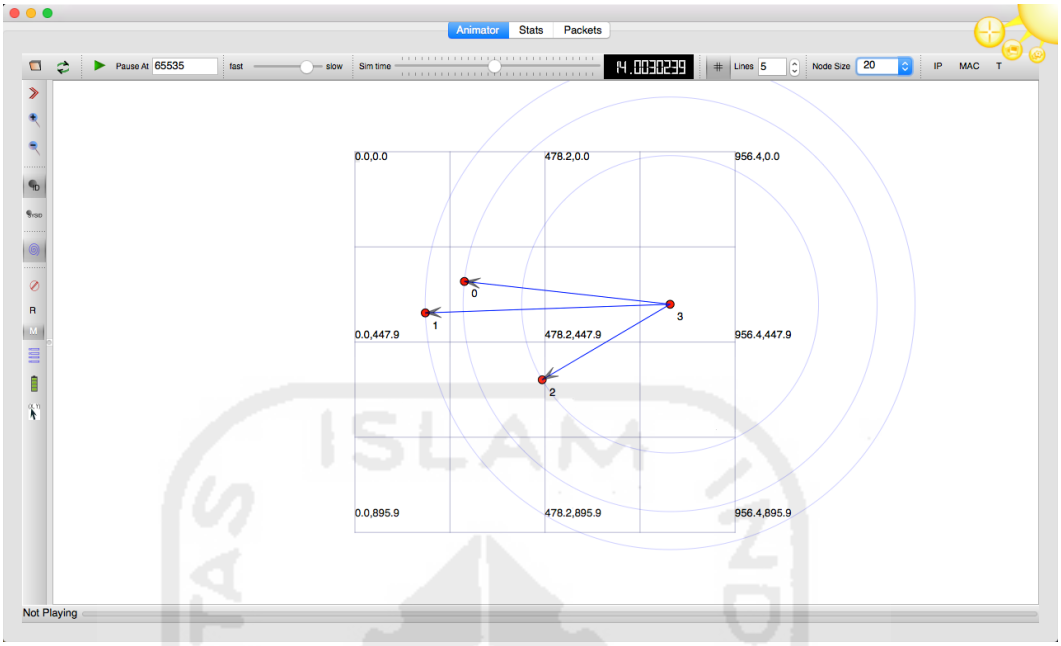
Untuk menghasilkan *file* dengan format *.flowmon*, masukkan perintah di bawah ini:

```
Ptr<FlowMonitor> flowmon;
FlowMonitorHelper flowmonHelper;
flowmon = flowmonHelper.InstallAll ();
flowmon->SerializeToXmlFile
(("nama_file.flowmon").c_str(), false, false);
```

Ketiga parameter tersebut antara lain adalah nama *file output*, nilai *boolean* untuk memanggil fungsi *enableHistograms,* dan nilai *boolean* untuk memanggil fungsi *enableProbes*.

c. *Output Trace File Format .xml*

File *output* dengan format *.xml* merupakan file untuk menampilkan pergerakan *node-node* dalam bentuk animasi yang dieksekusi oleh aplikasi NetAnim. Ketika file dieksekusi, simulasi akan berjalan sesuai dengan waktu yang telah ditentukan seperti ditunjukkan pada Gambar 3.4.



Gambar 3.5 Eksekusi File .xml

Untuk menghasilkan *file* dengan format *.xml* yang akan menjadi file animasi, ketikkan perintah di bawah ini:

```
#include "ns3/netanim-module.h"
AnimationInterface anim ("manet-olsr.xml");
  anim.SetMobilityPollInterval        (Seconds
  (0.2));
```

Nilai 0.2 merupakan interval waktu untuk tiap node mengambil informasi tentang posisi node lain.

3.4. Instalasi NS-3

Pada tugas akhir ini, penulis membutuhkan beberapa perangkat lunak untuk mendukung simulasi jaringan agar mendapatkan hasil yang maksimal. Perangkat-perangkat lunak tersebut antara lain adalah NS-3 dan NetAnime sebagai animator serta Ubuntu sebagai sistem operasinya.

a. Langkah-langkah diperlukan untuk instalasi NS-3:

1. Sebelum melakukan instalasi NS-3, perlu di install *package-package* yang menjadi pendukung performa NS-3, maka perlu dieksekusi perintah:

```
sudo apt-get install gcc g++ python python-dev
mercurial bzr gdb valgrind gsl-bin libgsl0-dev
libgsl0ldbl flex bison tcpdump sqlite sqlite3
libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0
libgtk2.0-dev uncrustify doxygen graphviz
imagemagick texlive texlive-latex-extra texlive-
generic-extra texlive-generic-recommended
texinfo dia texlive texlive-latex-extra texlive-
extra-utils texlive-generic-recommended
texi2html python-pygraphviz python-kiwi python-
pygoocanvas libgoocanvas-dev python-pygccxml
```

2. Setelah *package* pendukung telah ter *install*, maka perlu mengunduh *source code* NS-3 dengan perintah:

```
wget http://www.nsnam.org/release/ns-allinone-
3.24.1.tar.bz2
tar xjf ns-allinone-3.24.1.tar.bz2
cd ns-allinone-3.24.1/
```

3. Setelah *source code* NS-3 telah diunduh, maka perlu diekseskusi perintah berikut untuk *build examples:*

```
./build.py --enable-examples --enable-tests
```

4. Agar sistem dapat terkonfigurasi dengan waf (*build tool*) maka perlu dieksekusi perintah berikut:

```
./waf -d debug --enable-examples --enable-tests
configure
```

5. Berikut adalah perintah untuk mengeksekusi *file*:

```
./waf --run nama_file
```

b. Langkah-langkah instalasi NetAnim:

1. Sebelum melakukan instalasi NetAim, diperlukan beberapa *package* pendukung, maka perlu dieksekusi perintah sebagai berikut:

```
sudo apt-get download qt4-qmake libqt4-dev
```

```
libxml2-dev
```

2. Kemudian untuk masuk ke dalam folder NetAnim yang ada di dalam folder NS-3, perlu dieksesusi perintah:

```
cd netanim-3.106/
```

3. Untuk menjalankan proses instalasi maka perlu dieksekusi perintah:

```
make clean
qmake NetAnim.pro
make
```

4. Untuk menjalankan NetAnim, perlu dieksekusi perintah:

```
./NetAnim
```

3.5. Instalasi OLSR pada *Node*

Langkah-langkah instalasi OLSR pada linux adalah sebagai berikut:

1. Perlu mengunduh beberapa *package* pendukung dengan perintah:

```
sudo apt-get install kernel-package libncurses5-dev fakeroot wget bzip2 g++ libssl-dev libxml2-dev doxygen bison flex libc6
```

2. Perlu mengunduh *source code* OLSR dari alamat http://www.olsr.org/?q=download
3. Setelah *file* OLSR terunduh, maka perlu melakukan *copy, extract,* dan *compile file* tersebut dengan perintah:

```
cp olsrd-0.6.3.tar.bz2 /usr/local/src
cd /usr/local/src
tar jxvf olsrd-0.6.3.tar.bz2
cd /usr/local/src/olsrd-0.6.3
make all
make install
make libs
make install_libs
mkdir -p /etc
cp -i files/olsrd.conf.default.lq
```

```
/etc/olsrd.conf
```

4. Sebelum menjalankan OLSR, pastikan *interface* yang ingin digunakan sebagai OLSR telah tercantum pada file `/etc/olsrd.conf` :

   ```
   sudo nano /etc/olsrd.conf
   ```

5. OLSR dapat dijalankan dengan perintah:

   ```
   sudo /etc/init.d/network restart
   sudo /etc/init.d/olsrd start
   ```