BAB IV
HASIL DAN PEMBAHASAN

4.1 Implementasi

Dalam implementasi ini akan memaparkan mengenai beberapa tahapan yang dilakukan selama proses pembuatan simulasi. Fokus penelitian ini terkait dengan evalusasi performansi *routing protocol* OLSR dan AODV yang diterapkan pada jaringan VANET. Untuk itu beberapa hal yang dilakukan di antaranya dengan membuat program simulasi, menjalankan program simulasi sesuai skenario yang telah ditentukan dan mengumpukan data yang dihasilkan.

Dalam pembuatan program simulasi, penulis memanfaatkan beberapa referensi kode program yang terdapat pada direktori NS-3 yaitu *wave-simple-80211p*, *manet routing compare* dan modul *library* lainnya yang akan ditambahkan baris kode sesuai kebutuhan skenario. Hal tersebut berguna untuk mendukung selama proses simulasi jaringan VANET.

Setelah membuat program simulasi sesuai yang dibutuhkan, kemudian menjalankan program simulasi dengan skenario yang telah ditentukan. Beberapa *output file* yang dihasilkan selama proses simulasi berlangsung berguna untuk pengambilan data dan mendukung analisa kinerja dari *routing protocol* OLSR dan AODV.

4.1.1 Kode Program Simulasi VANET

a. Memanggil *Library* Pada NS-3

Sebelum membuat program simulasi perlu untuk mendeklarasikan *library* terlebih dahulu. *Library* merupakan kumpulan beberapa fungsi atau program yang telah tersedia sehingga akan memudahkan dalam memanggil fungsi. Dalam NS-3 terdapat modul *library* yang dapat dipanggil guna mendukung program simulasi. Dalam hal ini, penulis menggunakan modul *library* seperti terlihat pda gambar Gambar 4.1.

```
 1  #include <fstream>
 2  #include <iostream>
 3  #include "ns3/core-module.h"
 4  #include "ns3/network-module.h"
 5  #include "ns3/internet-module.h"
 6  #include "ns3/mobility-module.h"
 7  #include "ns3/aodv-module.h"
 8  #include "ns3/olsr-module.h"
 9  #include "ns3/applications-module.h"
10  #include "ns3/animation-interface.h"
11  #include "ns3/flow-monitor-module.h"
12  #include "ns3/wifi-80211p-helper.h"
```

```
#include "ns3/wave-mac-helper.h"
#include "ns3/ocb-wifi-mac.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/gauss-markov-mobility-model.h"
#include "ns3/log.h"
#include "ns3/string.h"
#include "ns3/socket.h"
#include "ns3/double.h"
#include "ns3/config.h"
#include "ns3/log.h"
#include "ns3/command-line.h"
```

Gambar 4.1 Baris Kode *Modul Library*

b. Fungsi Utama

Fungsi utama berkaitan dengan inisialisai kelas, pembuatan *file .csv* dan perintah menjalankan beberapa fungsi awal sampai akhir simulasi, seperti terlihat pada gambar Gambar 4.2.

```
int
main (int argc, char *argv[])
{
VanetRoutingExperiment experiment;
std::string CSVfileName = experiment.CommandSetup (argc,argv);
std::ofstream out (CSVfileName.c_str ());
out << "Detik Simulasi," <<
"Receive Rate," <<
"Packets Received," <<
"Jumlah Koneksi," <<
"Jenis Routing Protocol," <<
"Transmission Power" <<
std::endl;
out.close ();
double txp = 15;
int nodeSinks =8;
experiment.Run (nodeSinks, txp, CSVfileName);
}
```

Gambar 4.2 Baris Kode Fungsi Utama

c. Inisialisasi Kelas Utama

Kelas utama ini berisi beberapa variable dan fungsi-fungsi yang akan dijalankan, seperti terlihat pada Gambar 4.3.

```
class VanetRoutingExperiment
{
public:
VanetRoutingExperiment ();
void Run (int nodeSinks, double txp, std::string CSVfileName);
std::string CommandSetup (int argc, char **argv);
private:
Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
void ReceivePacket (Ptr<Socket> socket);
void CheckThroughput ();
std::string m_CSVfileName;
```

```
12  std::string m_protocolName;
13  std::string packetSize;
14  uint32_t port;
15  uint32_t bytesTotal;
16  uint32_t packetsReceived;
17  int m_nodeSinks;
18  double m_txp;
19  uint32_t m_nodeWifis;
20  uint32_t m_speed;
21  uint32_t m_protocol;
22  bool verbose = false;
23  };
24  VanetRoutingExperiment::VanetRoutingExperiment ()
25  :port (9),
26  bytesTotal (0),
27  packetsReceived (0),
28  m_CSVfileName (".csv"),
29  packetSize (""),
30  m_protocol (0)
```

Gambar 4.3 Baris Kode Kelas Utama

d. Print Received Packet

Kelas ini berfungsi untuk mencetak paket data yang dikirim dan paket data yang diterima, seperti terlihat pada Gambar 4.4.

```
1   static inline std::string
2   PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet, Address
3   senderAddress)
4   {
5   std::ostringstream oss;
6   oss << Simulator::Now ().GetSeconds () << " node " << socket-
7   >GetNode()->GetId ();
8   if (InetSocketAddress::IsMatchingType (senderAddress))
9   {
10  InetSocketAddress                addr=InetSocketAddress::ConvertFrom
11  (senderAddress);
12  oss << " menerima satu paket dari " << addr.GetIpv4 ();
13  }
14  else
15  {
16  oss << " menerima satu paket!";
17  }
18  return oss.str ();
19  }
20  void
21  VanetRoutingExperiment::ReceivePacket (Ptr<Socket> socket)
22  {
23  Ptr<Packet> packet;
24  Address senderAddress;
25  while ((packet = socket->RecvFrom (senderAddress)))
26  {
27  bytesTotal += packet->GetSize ();
28  packetsReceived += 1;
29  NS_LOG_UNCOND (PrintReceivedPacket (socket, packet, senderAddress));
30  }
31  }
32  Ptr<Socket>
```

```
33 VanetRoutingExperiment::SetupPacketReceive(Ipv4Address          addr,
34 Ptr<Node> node)
35 {
36 TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
37 Ptr<Socket> nsink = Socket::CreateSocket (node, tid);
38 InetSocketAddress local = InetSocketAddress (addr, port);
39 nsink->Bind (local);
40 nsink->SetRecvCallback
41 (MakeCallback(&RoutingExperiment::ReceivePacket, this));
42 return nsink;
43 }
```

Gambar 4.4 Baris Kode *Print Received Packet*

e. Fungsi Untuk Menghitung *Throughput*

Kelas ini berfungsi untuk menghitung nilai *Throughput* dengan menghitung besarnya data yang diterima setiap detik, seperti terlihat pada Gambar 4.5.

```
1  void
2  VanetRoutingExperiment::CheckThroughput ()
3  {
4  double kbs = (bytesTotal * 8.0) / 1024;
5  bytesTotal = 0;
6  std::ofstream out (m_CSVfileName.c_str (), std::ios::app);
7  out << (Simulator::Now ()).GetSeconds () << ","
8  << kbs << ","
9  << packetsReceived << ","
10 << m_nodeSinks << ","
11 << m_protocolName << ","
12 << m_txp << ""
13 << std::endl;
14 out.close ();
15 packetsReceived = 0;
16 Simulator::Schedule(Seconds(1.0),
17 &VanetRoutingExperiment::CheckThroughput, this);
18 }
```

Gambar 4.5 Baris Kode Menghitung *Throughput*

f. *Set Sink* Dan Node Sumber

Kelas ini berfungsi sebagai *traffic generator* lalu lintas data dan menentukan node sumber yang mengirimkan paket data ke node tujuan serta dapat memulai kapan paket data dikirimkan dan memberhentikan pengiriman paket data, seperti terlihat pada Gambar 4.6.

```
1  OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
2  onoff1.SetAttribute("OnTime",StringValue("ns3::ConstantRandomVariabl
3  e[Constant=1.0]"));
4  onoff1.SetAttribute("OffTime",StringValue("ns3::ConstantRandomVariab
5  le[Constant=0.0]"));
6  for (int i = 0; i < nodeSinks; i++)
7  {
8  Ptr<Socket> nsink = SetupPacketReceive (adhocInterfaces.GetAddress
9  (i), adhocNodes.Get (i));
10 AddressValue              remoteAddress              (InetSocketAddress
11 (adhocInterfaces.GetAddress (i), port));
```

```
onoff1.SetAttribute ("Remote", remoteAddress);
Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable>
();
ApplicationContainer temp = onoff1.Install (adhocNodes.Get (i +
nodeSinks));
temp.Start (Seconds (var->GetValue (1.0,2.0)));
temp.Stop (Seconds (TotalTime));
}
```

Gambar 4.6 Baris Kode Fungsi *Sink* Dan Node Sumber

g. Command Setup

Kelas ini berfungsi untuk menyimpan nilai parameter yang telah dibuat dan akan di *input* ketika menjalankan program pada konsol *terminal*, seperti terlihat pada Gambar 4.7.

```
std::string
VanetRoutingExperiment::CommandSetup (int argc, char **argv)
{
CommandLine cmd;
cmd.AddValue ("CSVfileName", "The name of the CSV output file name",
m_CSVfileName);
cmd.AddValue ("protocol", "1=OLSR;2=AODV", m_protocol);
cmd.AddValue ("nodeTotal", "Total Node", m_nodeWifis);
cmd.AddValue ("speed", "Speed Node", m_speed);
cmd.AddValue ("verbose", "turn on all WifiNetDevice log components",
verbose);
cmd.Parse (argc, argv);
return m_CSVfileName;
}
```

Gambar 4.7 Baris Kode Fungsi *Command Setup*

h. Membuat Node

Fungsi ini berguna untuk membuat node dalam simulasi atau mewakili sebuah perangkat *computer*, seperti terlihat pada Gambar 4.8.

```
NodeContainer adhocNodes;
adhocNodes.Create (nodeWifis);
```

Gambar 4.8 Baris Kode Fungsi Membuat Node

i. Inisialisasi Dan Pengaturan *Wifi*

Inisialisasi pengaturan *wifi channel* dan tipe *wifi* yang dipergunakan. Pada tipe *wifi* ini berfungsi sebagai sarana pengirim dan penerima data pada sebuah node. Selain itu mengatur model propagasi radio yang digunakan, seperti terlihat pada Gambar 4.9.

```
YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
Config::SetDefault
("ns3::WifiRemoteStationManager::RtsCtsThreshold",       StringValue
("20"));
YansWifiChannelHelper wifiChannel;

```

```
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelay
Model");
wifiChannel.AddPropagationLoss("ns3::ThreeLogDistancePropagationLos
sModel");
Ptr<YansWifiChannel> wifiChannelPtr = wifiChannel.Create ();
wifiPhy.SetChannel (wifiChannelPtr);
NqosWaveMacHelper wifi80211pMac = NqosWaveMacHelper::Default();
Wifi80211pHelper wifi80211p = Wifi80211pHelper::Default ();
//wifi80211p.SetStandard (WIFI_PHY_STANDARD_80211_10MHZ);

wifi80211p.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                                    "DataMode",StringValue (phyMode),
"ControlMode",StringValue(phyMode));
 if (verbose)
    {
     wifi80211p.EnableLogComponents (); // Turn on all Wifi 802.11p
logging
    }
```

Gambar 4.9 Baris Kode Fungsi Inisialisai Dan Pengaturan *Wifi*

j. *Transmission Power*

Transmission power yaitu besaran daya yang digunakan serta menentukan jarak trasmisi dari jangkauan *wireless,* seperti terlihat pada Gambar 4.10.

```
wifiPhy.Set ("TxPowerStart", DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
wifiPhy.Set ("RxGain", DoubleValue (-10));
```

Gambar 4.10 Baris Kode Fungsi *Transmission Power*

k. Membuat *Device Wifi Card*

Membuat *device* berfungsi untuk meng-*install* sebuah kartu *wireless card* pada setiap node yang digunakan untuk menghubungkan ke node lain, seperti terlihat pada Gambar 4.11.

```
wifi80211pMac.SetType ("ns3::OcbWifiMac");
NetDeviceContainer adhocDevices = wifi80211p.Install (wifiPhy,
wifi80211pMac, adhocNodes);
```

Gambar 4.11 Baris Kode Fungsi Membuat *Wifi Card*

l. Membuat Area Simualsi

Area simulasi berfungsi sebagai luas wilayah yang diguanakan untuk melakukan simulasi. Selain itu untuk mengatur bagaimana suatu node dideklarasikan dari posisi awal, seperti terlihat pada Gambar 4.12.

```
1  MobilityHelper mobilityAdhoc;
2  mobilityAdhoc.SetPositionAllocator
3  ("ns3::RandomBoxPositionAllocator","X",StringValue
4  ("ns3::UniformRandomVariable[Min=0|Max=1000]"),"Y",StringValue
5  ("ns3::UniformRandomVariable[Min=0|Max=1000]"),"Z",StringValue
6  ("ns3::UniformRandomVariable[Min=0|Max=100]"));
```

Gambar 4.12 Baris Kode Fungsi Membuat Area Simulasi

m. *Set* Pergerakan Node Dan Kecepatan

Pada bagian ini berfungsi untuk membuat jenis model pergerakan dari suatu node dan menentukan kecepatan dari suatu node, seperti terlihat pada Gambar 4.13.

```
1  mobilityAdhoc.SetMobilityModel ("ns3::GaussMarkovMobilityModel",
2  "Bounds", BoxValue (Box (0, 1000, 0, 1000, 0, 100)),
3  "TimeStep", TimeValue (Seconds (1.0)),
4  "MeanVelocity", StringValue (ssSpeed.str ()));
5  std::stringstream ssSpeed;
6  ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed <<
7  "]";
8  //ssSpeed << "ns3::ConstantRandomVariable[Constant=" << nodeSpeed <<
9  "]";
10 mobilityAdhoc.Install (adhocNodes);
11 streamIndex += mobilityAdhoc.AssignStreams (adhocNodes, streamIndex);
```

Gambar 4.13 Baris Kode Fungsi Membuat Pergerakan Dan Kecepatan Node

n. *Set Routing Protocol*

Pada bagian ini berfungsi sebagai inisialisasi dan pemilihan jenis *routing protocol* yang akan digunakan pada saat simulasi, seperti terlihat pada Gambar 4.14.

```
1  AodvHelper aodv;
2  OlsrHelper olsr;
3  Ipv4ListRoutingHelper list;
4  InternetStackHelper internet;
5  switch (m_protocol)
6  {
7  case 1:
8  list.Add (olsr, 100);
9  m_protocolName = "OLSR";
10 internet.SetRoutingHelper (list);
11 internet.Install (adhocNodes);
12 break;
13 case 2:
14 list.Add (aodv, 100);
15 m_protocolName = "AODV";
16 internet.SetRoutingHelper (list);
17 internet.Install (adhocNodes);
```

```
18  break;
19  default:
20  NS_FATAL_ERROR ("No such protocol:" << m_protocol);
21  }
```

Gambar 4.14 Baris Kode Fungsi Membuat *Routing Protokol*

o. *Set* IP *Address*

Pada bagian ini berfungsi untuk membuat alamat IP pada setiap node dan menentukan rentang alamat IP yang digunakan, seperti terlihat pada Gambar 4.15.

```
1  Ipv4AddressHelper addressAdhoc;
2  addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
3  Ipv4InterfaceContainer adhocInterfaces;
4  adhocInterfaces = addressAdhoc.Assign (adhocDevices);
```

Gambar 4.15 Baris Kode Fungsi Membuat IP *Address*

p. *Print* Paket Menggunakan *Flowmonitor*

Pada bagian ini berfungsi sebagai data *collection* pada saat simulasi berlangsung. Pada saat simulasi dijalankan *flowmonitor* akan menghitung pengiriman paket, *Delay*, jumlah paket yang dikirimkan dan diterima, jumlah paket yang hilang dan waktu pengiriman, seperti terlihat pada Gambar 4.16.

```
1  Ptr<FlowMonitor> flowMonitor;
2  FlowMonitorHelper flowmonHelper;
3  flowMonitor = flowmonHelper.InstallAll ();
4  flowMonitor->SerializeToXmlFile("vanet-flowmon.xml",true,true);
```

Gambar 4.16 Baris Kode Fungsi *Print* Paket *Dengan Flowmonitor*

q. Simulasi Dimulai Dan Berhenti

Pada bagian ini berfungsi untuk mengawali dan memberhentikan simulasi, seperti terlihat pada Gambar 4.17.

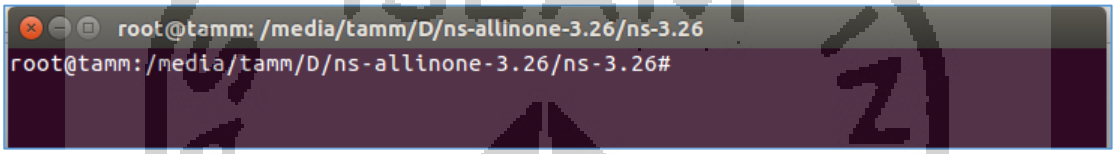
```
1  Simulator::Stop (Seconds (TotalTime));
2  Simulator::Run ();
3  Simulator::Destroy ();
```

Gambar 4.17 Baris Kode Fungsi Simulasi Dimulai Dan Berhenti

4.1.2 Tahapan Menjalankan Simulasi

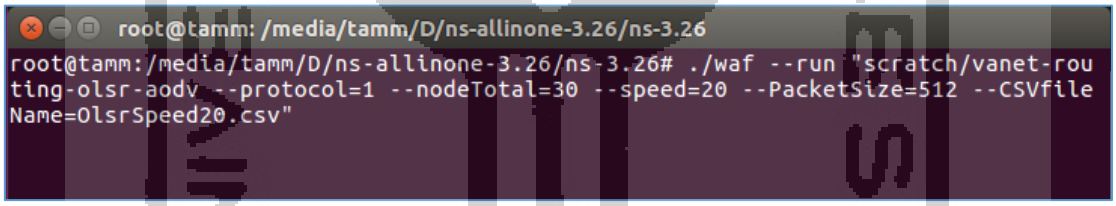
Pada tahapan ini akan dilakukan proses menjalankan simulasi terhadap program yang telah dibuat dengan mengeksekusi perintah tertentu. Perintah tersebut akan disesuaikan dengan skenario yang telah dibuat. Adapun langkah-langkah dalam menjalankan simulasi sebagai berikut:

a. Langkah awal untuk menjalankan simulasi program, pengguna perlu masuk ke dalam direktori *file* NS-3 berada pada direktori ns-allinone 3.26 melalui konsol *terminal* yang terdapat *file* dengan nama *waf* melalui, seperti terlihat pada Gambar 4.18.



Gambar 4.18 Direktori *File* NS-3

b. Setelah masuk ke dalam direktori file NS-3, selanjutnya pengguna perlu mengetik baris perintah pada konsol *terminal* untuk menjalankan program simulasi, seperti terlihat pada Gambar 4.19.

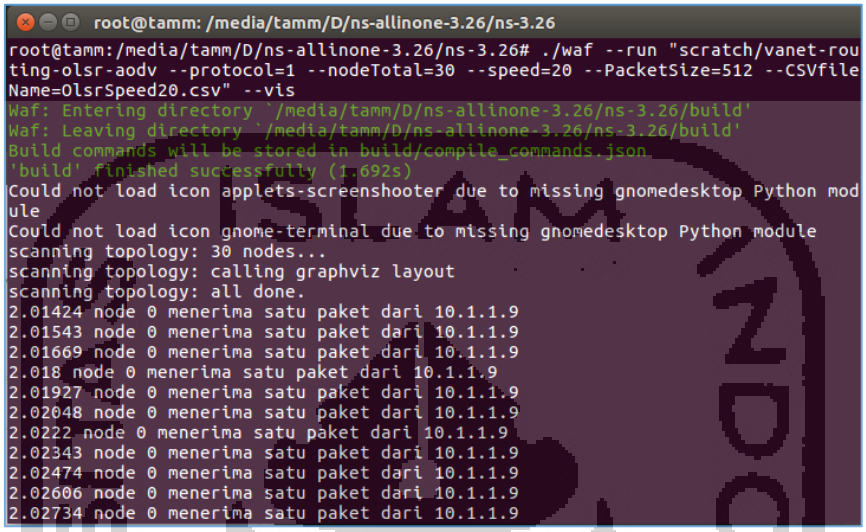


Gambar 4.19 Baris Perintah Menjalankan Simulasi

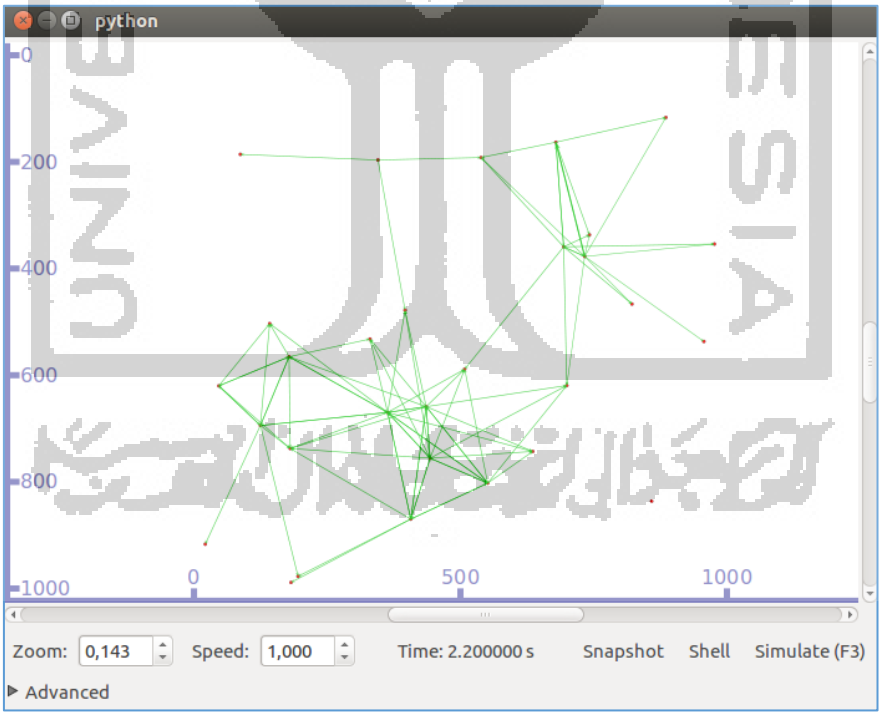
Penjelasasan:

./waf -- run	: perintah dasar untuk menjalankan program simulasi.
scratch	: direktori program disimpan.
vanet-routing olsr-aodv	: nama file program simulasi.
protocol	: jenis *protocol* yang dipilih.
nodeTotal	: jumlah node yang digunakan.
PacketSize	: ukuran paket data yang dikirimkan.
speed	: ukuran kecepatan node yang digunakan.
CSVfileName	: nama untuk menyimpan file *.csv.*
--vis	: perintah untuk menampilkan proses visualisasi simulasi.

c. Kemudian setelah menjalankan program simulasi sesuai dengan skenario yang telah ditentukan, maka NS-3 akan melakukan proses *build* dan *compile* terhadap program simulasi yang telah dibuat dan akan ditampilkan seperti terlihat pada Gambar 4.20 dan Gambar 4.21.



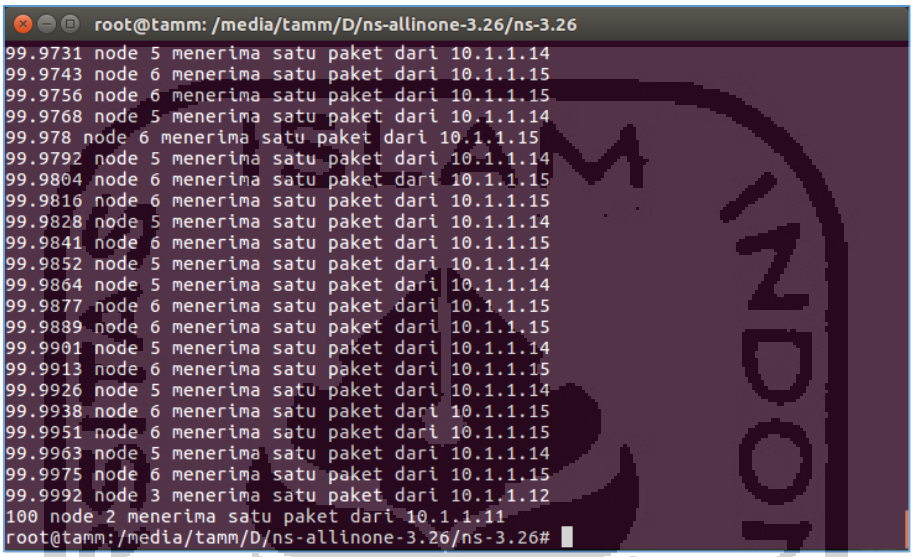
Gambar 4.20 Hasil *Running* Program Simulasi



Gambar 4.21 Visualisi Simulasi Jaringan

Pada Gambar 4.20 merupakan *capture* dari proses pengiriman paket data dari node sumber ke node tujuan dan Gambar 4.21 menunjukan proses visualisai simulasi jaringan yang dijalankan secara langsung.

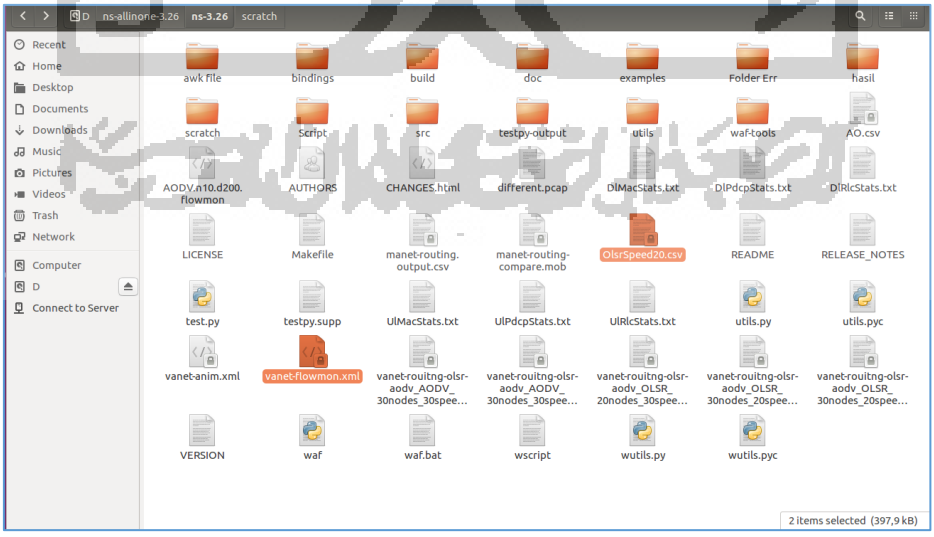
d. Kemudian proses simulasi akan berhenti dan berakhir. Seperti terlihat pada Gambar 4.22



Gambar 4.22 Tampilan Program Simulasi Berakhir

4.1.3 Tahapan Pengambilan Data

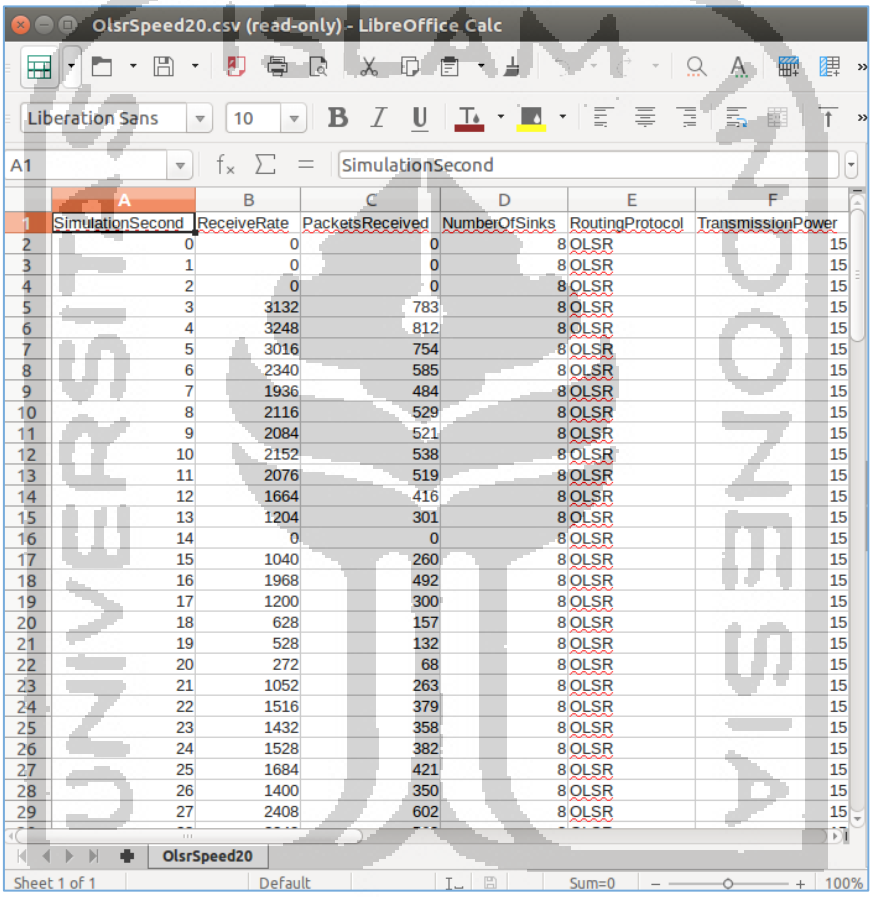
Pada tahapan ini dilakukan pengambilan data setelah proses simulasi berakhir. Selama melakukan proses simulasi akan menghasilkan *file output* hasil simulasi, yaitu *file* dengan ekstensi *.csv* dan file dengan ekstensi *flowmon.xml*, seperti terlihat pada Gambar 4.23



Gambar 4.23 *Output File* Hasil Simulasi

1. *File .csv*

Pada *file .csv* terdapat beberapa informasi selama proses simulasi berlangsung, informasi tersebut mengenai waktu simulasi, laju data diterima, paket data yang diterima, jumlah node yang mengirimakan data, jenis *protocol* yang digunakan dan transmisi daya nirkabel. Adapun informasi tersebut digunakan untuk menghitung nilai rata-rata *Throughput* yang dihasilkan selama proses simulasi berlangsung dengan menghitung total paket diterima dan waktu yang dibutuhkan selama proses pengiriman data, seperti terlihat pada Gambar 4.24.



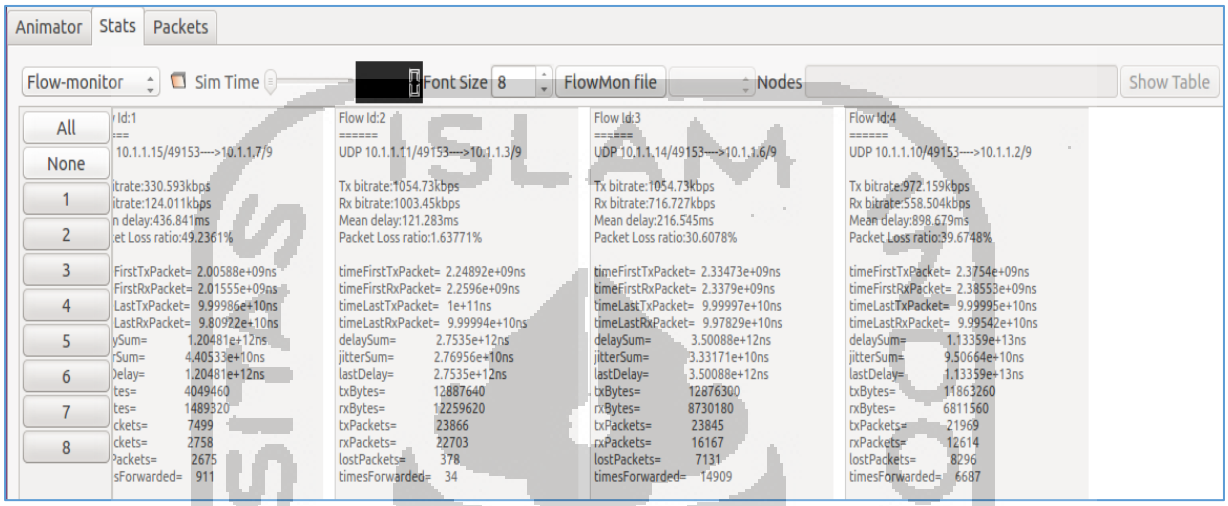
SimulationSecond	ReceiveRate	PacketsReceived	NumberOfSinks	RoutingProtocol	TransmissionPower
0	0	0	8	OLSR	15
1	0	0	8	OLSR	15
2	0	0	8	OLSR	15
3	3132	783	8	OLSR	15
4	3248	812	8	OLSR	15
5	3016	754	8	OLSR	15
6	2340	585	8	OLSR	15
7	1936	484	8	OLSR	15
8	2116	529	8	OLSR	15
9	2084	521	8	OLSR	15
10	2152	538	8	OLSR	15
11	2076	519	8	OLSR	15
12	1664	416	8	OLSR	15
13	1204	301	8	OLSR	15
14	0	0	8	OLSR	15
15	1040	260	8	OLSR	15
16	1968	492	8	OLSR	15
17	1200	300	8	OLSR	15
18	628	157	8	OLSR	15
19	528	132	8	OLSR	15
20	272	68	8	OLSR	15
21	1052	263	8	OLSR	15
22	1516	379	8	OLSR	15
23	1432	358	8	OLSR	15
24	1528	382	8	OLSR	15
25	1684	421	8	OLSR	15
26	1400	350	8	OLSR	15
27	2408	602	8	OLSR	15

Gambar 4.24 Hasil *File Csv*

2. *File Flowmon.xml*

Pada *file flowmon.xml* merupakan *flow monitor* atau grafik dari node yang mengirimkan paket data ke node tujuan. *File flowmon.xml* dapat dibuka menggunakan *software* Netanim yang dapat dipilih dari menu *Stats*. Di dalam *file flow monitor* tersebut terdapat beberapa informasi seperti *Delay*, *Paket Loss*, paket yang dikirim, *Jitter*, paket yang diterima dan lainnya. Dari informasi tersebut dapat dipergunakan untuk menghitung rata-rata nilai dari *Packet Loss Rate*, *Packet Delivery Ratio*, *Delay* dan *Jitter*. Untuk menghitung nilai rata-rata *Delay* dan *Jitter* selama proses simulasi dapat dilihat *mean delay* dan *JitterSum* setiap node

pengirim dan penerima serta melakukan penjumlahan dari *mean delay* dan *Jitter* dibagi dengan jumlah node penerima yang telah ditentukan. Sedangkan untuk menghitung nilai rata-rata *Packet Loss Rate dan Packet Delivery Ratio* selama proses simulasi dapat dilihat pada *txPacket* dan *rxPacket* sebagai paket yang dikirim dan paket yang diterima. Adapun isi dari *file flowmon* terlihat pada Gambar 4.25.



Gambar 4.25 Hasil *File Flowmon.xml*

4.2 Analisa Data

Analisa data merupakan tahapan yang dilakukan untuk mengetahui perbandingan performansi dari *routing protocol* OLSR dan AODV berdasarkan parameter QoS dan skenario yang telah dibuat. Pengambilan analisa data menggunakan *file .csv* dan *flowmon.xml* yang telah dihasilkan pada saat simulasi. Adapun pengambilan data dapat dilakukan, di antaranya:

1. *Throughput*

   Pengambilan data *throughput* pada *file .csv* dilakukan dengan cara menghitung total *received rate* yang dihasilkan dibagi waktu yang dibutuhkan selama proses awal paket data dikirim hingga akhir paket data diterima.

2. *Packet Delivery Ratio*

   Pengambilan data PDR pada *file flowmon.xml* dilakukan dengan cara membagi *total rxPacket* dengan *total txPacket* kemudian dikalikan dengan 100% untuk mendapatkan bentuk persen.

3. *Packet Loss Ratio*

   Pengambilan data PLR pada *file flowmon.xml* dilakukan dengan cara mengurangi *total txPacket* dengan *total rxPacket* kemudian hasilnya dibagi dengan *total txPacket* dan dikalikan 100% untuk mendapatkan bentuk persen.

4. *Delay*

   Pengambilan data rata-rata *Delay* pada *file flowmon.xml* dilakukan dengan cara melihat pada *Mean Delay* kemudian menjumlahkan *total Mean Delay* dibagi dengan banyaknya jumlah koneksi.

5. *Jitter*

   Pengambilan data rata-rata *Jitter* pada *file flowmon.xml* dilakukan dengan cara melihat pada *JitterSum* kemudian menjumlahkan *total JitterSum* dibagi dengan banyaknya jumlah koneksi.

4.2.1 Hasil Simulasi Skenario Penambahan Node

Pada simulasi skenario penambahan node dilakukan penambahan jumlah node dari 30 node, 50 node dan 80 node dengan area simulasi 1 km². Kecepatan yang digunakan untuk setiap node bervariasi antara 0-20 m/s dengan pergerakan secara acak. Kemudian hasil analisa performasi *routing protocol* OLSR dan AODV akan diukur berdasarkan parameter QoS yang dibagi dalam 4 bagian.

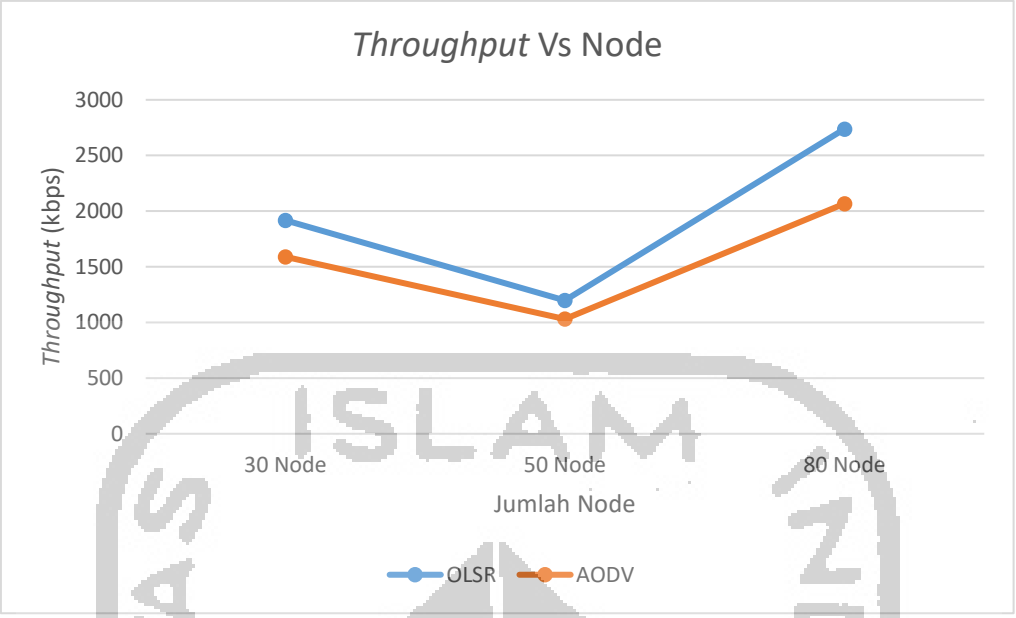
1. *Throughput*

Analisa dari hasil perhitungan rata-rata *Throughput* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario penambahan node dapat terlihat pada Tabel 4.1. Adapun satuan yang digunakan yaitu kbps *(kilo bit per second).*

Tabel 4.1 Nilai Rata-Rata *Throughput* Pada Skenario Penambahan Node

Protocol	**30 Node**	**50 Node**	**80 Node**	**Rata-Rata**
OLSR	1917,48	1197,93	2738,22	1951,21
AODV	1590,81	1030,44	2067,14	1562,80

Dari data Tabel 4.1 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata *Throughput* dari skenario penambahan node yang terlihat pada Gambar 4.26.



Gambar 4.26 Grafik *Throughput* Vs Node

Berdasarkan grafik yang terlihat pada Gambar 4.26 menunjukkan bahwa nilai *Throughput* pada node 30 *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih 326,66 kbps. Namun pada penambahan jumlah node 50, kedua *protocol* mengalami penurunan yaitu *protocol* OLSR menurun nilai *Throughput* sebesar 719,54 kbps dari 1917,48 kbps menjadi 1197,93 kbps dan *protocol* AODV sebesar 560,367 kbps dari 1590,81 kbps menjadi 1030,44 kbps dengan selisih kedunaya sebesar 167,48 kbps. Pada kepadatan yang meningkat yaitu 80 node, *protocol* OLSR maupun *protocol* AODV mampu meningkat signifikan nilai *Throughput* sebesar 1540,28 kbps dari 1197,93 kbps menjadi 2738,22 kbps untuk *protocol* OLSR dan *protocol* AODV sebesar 1036,69 kbps dari 1030,44 kbps menjadi 2067,14 kbps dengan selisih di antara keduanya sebesar 671,08 kbps. Sehingga nilai rata-rata *Throughput* dari *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai 1951,21 kbps dan 1562,80 kbps dengan selisih 388,41 kbps terhadap skenario penambahan jumlah node.

Pada Gambar 4.26 menunjukkan perbandingan nilai *Throughput* antara *protocol* OLSR dan *protocol* AODV dengan skenario penambahan jumlah node 30, 50, dan 80. Pada saat kedua *protocol* tersebut mengalami penambahan jumlah node dari 30 menjadi 50 node mengalami penurunan nilai *Throughput*, hal itu karena pengaruh dari perubahan topologi yang menyebabkan node sumber dalam mengirimkan paket data ke node tujuan semakin jauh jarak antar node sehingga akan mengalami putusnya rute. Namun seiring bertambahnya node dari

50 menjadi 80 node, kedua *protocol* mengalami peningkatan nilai *Throughput.* Pada saat itu node sumber dalam mengirimkan paket data ke node tujuan dapat mempertahankan rute karena terjadi penambahan node yang dijadikan *hop* untuk meneruskan paket data ke node tujuan.

Jika dilihat dari Gambar 4.26 menunjukkan grafik nilai *Throughput* dari *protocol* OLSR lebih baik daripada *protocol* AODV. Hal tersebut karena pada *routing protocol* OLSR dalam mengirimkan paket data dari node sumber ke node tujuan menggunakan mekanisme *Multi Point Relay* (MPR) untuk mengurangi pesan *broadcast* yang memiliki informasi rute yang sama dan disimpan dalam tabel *routing*. Jadi node MPR saja yang dapat meneruskan pesan *broadcast* ke node tujuan sehingga dapat menurunkan penggunan *bandwidth.*

Sedangkan pada *routing protocol* AODV dalam mengirimkan paket data dari node sumber ke node tujuan dengan melakukan *route discovery* pada setiap node yang disimpan dalam tabel *routing* untuk setiap tujuan. Dalam proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk menemukan rute ke node tujuan. Apabila node menerima pesan RREQ yang memiliki informasi rute ke node tujuan maka akan mengirimkan pesan RREP sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Sehingga akan berdampak pada penggunaan *bandwidth* yang berlebihan. Namun kedua protokol mengalami penurunan dan peningkatan nilai *Througput* karena pengaruh perubahan topologi jaringan dan banyaknya *hop* yang dilalui sehingga dapat merubah rute dari node sumber ke node tujuan dalam mengirimkan paket data.

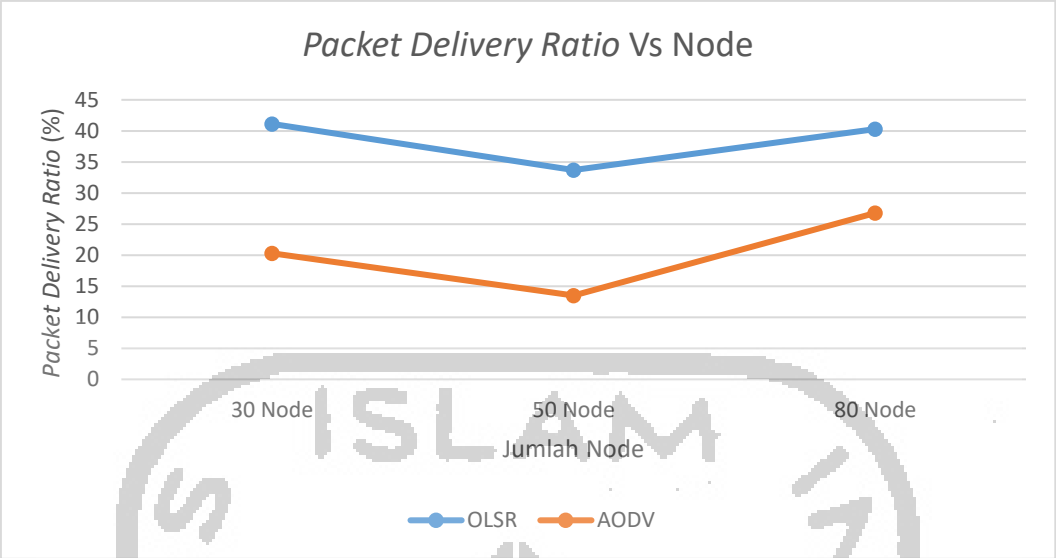
2. *Packet Delivery Ratio*

Analisa dari hasil perhitungan rata-rata *Packet Delivery Ratio* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario penambahan node dapat dilihat pada Tabel 4.2. Adapun satuan yang digunakan yaitu persentase.

Tabel 4.2 Nilai Rata-Rata PDR Pada Skenario Penambahan Node

Protocol	**30 Node**	**50 Node**	**80 Node**	**Rata-Rata**
OLSR	41,12	33,69	40,27	38,36
AODV	20,31	13,49	26,77	20,19

Dari data Tabel 4.2 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata PDR dari skenario penambahan node yang terlihat pada Gambar 4.27.



Gambar 4.27 Grafik *Packet Delivery Ratio* Vs Node

Berdasarkan grafik yang terlihat pada Gambar 4.27 menunjukkan bahwa persentase PDR pada node 30 *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih pesentase 20,80 %. Pada penambahan jumlah node 50, persentase PDR *protocol* OLSR menurun sebesar 7,42 % dari 41,12 % menjadi 33,69 % sedangkan *protocol* AODV menurun sebesar 6,82 % dari 20,31 % menjadi 13,49 % dengan selisih perbandingan antara keduanya sebesar 20,2 %. Pada kepadatan yang meningkat yaitu 80 node, *protocol* OLSR mampu meningkatkan persentase PDR sebesar 6,57 % dari 33,69 % menjadi 40,27 % dan *protocol* AODV meningkat sebesar 13,28 % dari 13,49 % menjadi 26,77 % dengan selisih sebesar 13,5 %. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata persentase 38,36 % dan 20,19 % dengan selisih sebesar 18,17 % terhadap skenario penambahan jumlah node.

Pada Gambar 4.27 menunjukkan perbandingan nilai *Packet Delivery Ratio* antara *protocol* OLSR dan *protocol* AODV dengan skenario penambahan jumlah node 30, 50, dan 80. Pada saat kedua *protocol* tersebut mengalami penambahan jumlah node dari 30 menjadi 50 node, *protocol* OLSR mengalami penurunan nilai *Packet Delivery Ratio*. Pada saat kedua *protocol* tersebut mengalami penambahan jumlah node dari 30 menjadi 50 node mengalami penurunan nilai *Packet Delivery Ratio*, hal itu karena pengaruh dari perubahan topologi yang menyebabkan node sumber dalam mengirimkan paket data ke node tujuan semakin jauh jarak antar node sehingga akan mengalami putusnya rute. Namun seiring bertambahnya node dari 50 menjadi 80 node terjadi peningkatan nilai *Packet Delivery Ratio* pada kedua *protocol*. Pada

saat itu node sumber dalam mengirimkan paket data ke node tujuan dapat mempertahankan rute karena terjadi penambahan node yang dijadikan *hop* untuk meneruskan paket data sehingga persentase penerimaan paket data ke node tujuan semakin meningkat.

Jika dilihat dari Gambar 4.27 menunjukkan grafik nilai *Packet Delivery Ratio* dari *protocol* OLSR lebih baik daripada *protocol* AODV. Hal tersebut karena pada *routing protocol* OLSR dalam mengirimkan paket data dari node sumber ke node tujuan dapat menyimpan informasi *routing* yang konsisten dan *up to date* pada setiap node sehingga dapat memberikan informasi rute dengan segera apabila diperlukan. Selain itu mekanisme MPR membuat proses pencarian rute menjadi efisien karena node MPR akan membuat paket OLSR yang diterima tidak akan langsung diteruskan ke node lain, tetapi hanya node yang dipilih sebagai MPR yang dapat menyampaikan paket kontrol yang diterima (Ainurrachman, Bhawiyuga, & Ichsan, 2017). Sehingga akan mempengaruhi tingkat keberhasilan paket yang dikirimkan. Sedangkan pada *routing protocol* AODV dalam mengirimkan paket data dari node sumber ke node tujuan dibuat oleh node sumber apabila membutuhkan informasi *routing* ke node tujuan. Dengan melakukan *route discovery* yang diminta dengan membanjiri jaringan untuk mencari rute dan selesai ketika rute ditemukan (Training, 2016). *Route discovery* pada setiap node disimpan dalam tabel *routing* untuk setiap tujuan. Proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk mengetahui informasi rute ke node tujuan. Apabila node menerima pesan RREQ yang memiliki informasi rute ke node tujuan maka akan mengirimkan pesan RREP (Rezkinanda & Anggoro, 2016). Sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Hal tersebut dapat mempengaruhi dalam mengirimkan paket data ke node tujuan karena akan menunggu *route discovery* selesai dilakukan.

Selain itu dengan peningkatan jumlah node akan membuat persentase PDR meningkat karena akan membentuk banyak *hop* yang dapat dilalui dan perubahan jalur topologi jaringan. Namun rendahnya pesentase PDR pada skenario penambahan jumlah node, karena dapat disebabkan beberapa hal, yaitu, tidak tersedianya rute karena node node sumber keluar dari jangkauan transmisi sinyal, perubahan topologi dan *congestion* paket data dalam jaringan.

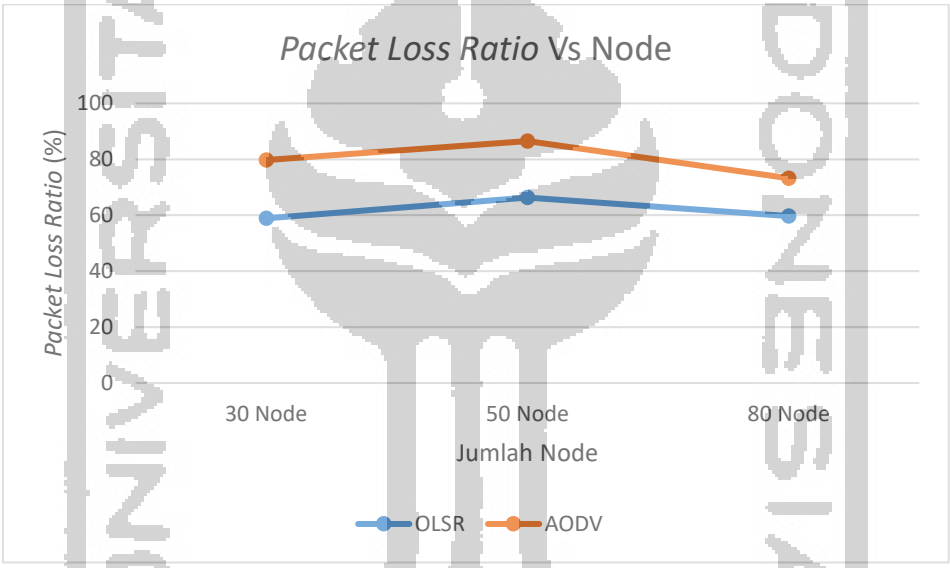
3. *Packet Loss Ratio*

Analisa dari hasil perhitungan rata-rata *Packet Loss Ratio* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario penambahan node dapat dilihat pada Tabel 4.3. Adapun satuan yang digunakan yaitu persentase.

Tabel 4.3 Nilai Rata-Rata PLR Pada Skenario Penambahan Node

Protocol	**30 Node**	**50 Node**	**80 Node**	**Rata-Rata**
OLSR	58,87	66,30	59,72	61,63
AODV	79,68	86,50	73,22	79,77

Dari data Tabel 4.3 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata PLR dari skenario penambahan node yang terlihat pada Gambar 4.28.



Gambar 4.28 Grafik *Packet Loss Ratio* Vs Node

Berdasarkan grafik yang terlihat pada Gambar 4.28 menunjukkan bahwa persentase PLR pada node 30 *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih pesentase 20,80 %. Pada penambahan jumlah node 50, persentase PLR *protocol* OLSR meningkat sebesar 7,42 % dari 58,87 % menjadi 66,30 % sedangkan *protocol* AODV juga mengalami peningkatan sebesar 6,9 % dari 79,68 % menjadi 86,50 %. Pada kepadatan yang meningkat yaitu 80 node, *protocol* OLSR mampu menurunkan tingkat persentase PLR sebesar 6,57 % dari 66,30 % menjadi 59,72 % daripada *protocol* AODV sebesar 13,28 % dari 86,50 % menjadi 73,22 %. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata persentase 61,63 % dan 79,77 % terhadap skenario penambahan jumlah node.

Dengan peningkatan jumlah node akan membuat persentase PLR menurun karena akan membentuk *multi-hop* yang dapat dilalui dan perubahan jalur topologi jaringan. Namun tingginya pesentase PLR pada skenario penambahan jumlah node, karena dapat disebabkan beberapa hal yaitu, tidak tersedianya rute karena node node sumber keluar dari jangkauan transmisi sinyal, perubahan topologi dan *congestion* paket data dalam jaringan.

Hasil dari *Packet Loss Ratio* (PLR) berbanding terbalik dengan hasil *Packet Delivery Ratio* (PDR) karena saling berkaitan satu sama lain, apabila nilai persentase dari PLR rendah atau menurun maka nilai persentase PDR tinggi atau meningkat dan sebaliknya. Karena pada nilai persentase PLR menggambarkan persentase paket hilang selama mengirimkan paket dan PDR menggambarkan persentase paket diterima oleh node tujuan.

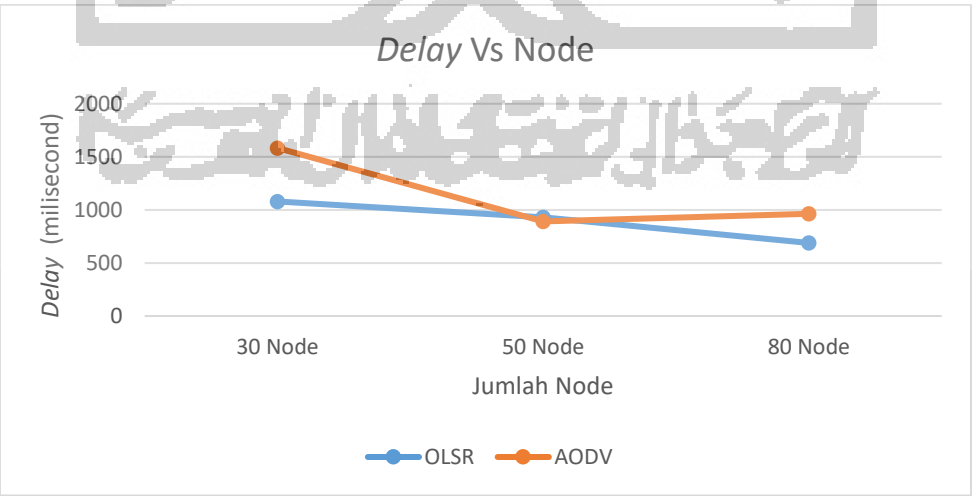
4. *Delay*

Analisa dari hasil perhitungan rata-rata *Delay* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario penambahan node dapat dilihat pada Tabel 4.4. Adapun satuan yang digunakan yaitu *milliseconds.*

Tabel 4.4 Nilai Rata-Rata *Delay* Pada Skenario Penambahan Node

Protocol	**30 Node**	**50 Node**	**80 Node**	**Rata-Rata**
OLSR	1079,04	928,87	688,64	898,85
AODV	1582,59	891,24	961,86	1145,23

Dari data Tabel 4.4 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata *Delay* dari skenario penambahan node yang terlihat pada Gambar 4.29.



Gambar 4.29 Grafik *Delay* Vs Node

Berdasarkan grafik yang terlihat pada Gambar 4.29 menunjukkan bahwa nilai *Delay* pada node 30 *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih *delay* 503,55 ms. Pada penambahan jumlah node 50, nilai *delay* kedua *protocol* mengalami penurunan yaitu *protocol* OLSR sebesar 150,17 ms dari 1079,04 ms menjadi 928,87 ms dan *protocol* AODV sebesar 691,35 ms dari 1582,59 ms menjadi 891,24 ms dengan selisih keduanya sebesar 37,63 ms. Seiring pada kepadatan yang meningkat yaitu 80 node, *protocol* OLSR mengalami penurunan nilai *Delay* sebesar 240,22 ms dari 928,87 ms menjadi 688,64 ms. Sedangkan *protocol* AODV meningkat sebesar 70,62 ms dari 891,24 ms menjadi 961,86 ms dengan selisih keduanya sebesar 273,22 ms. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata *Delay* 898,85 ms dan 1145,23 ms dengan selisih mencapai 246,38 ms terhadap skenario penambahan jumlah node.

Pada Gambar 4.29 menunjukkan perbandingan nilai *Delay* antara *protocol* OLSR dan *protocol* AODV dengan skenario penambahan jumlah node 30, 50, dan 80. Pada saat kedua *protocol* tersebut mengalami penambahan jumlah node dari 30 menjadi 50 node, kedua *protocol* mengalami penurunan nilai *Delay*, hal itu karena pengaruh dari perubahan topologi yang menyebabkan node sumber dalam mengirimkan paket data ke node tujuan semakin dekat jarak antar node sehingga memperkecil terputusnya rute dan pengiriman data lebih cepat. Seiring bertambahnya node dari 50 menjadi 80 node terjadi penurunan nilai *Delay* pada *protocol* OLSR sedangkan *protocol* AODV meningkat. Pada saat itu node sumber dalam mengirimkan paket data ke node tujuan dapat dengan cepat mengirimkan paket data, karena terjadi penambahan node yang dijadikan *hop* untuk meneruskan paket data ke node tujuan sehingga akan semakin cepat sampai. Sedangkan *protocol* AODV dipengaruhi oleh proses *route discovery* yang membuat *Delay* menjadi meningkat, karena banyaknya *hop* yang dilalui.

Jika dilihat dari Gambar 4.29 menunjukkan grafik nilai *Delay* dari *protocol* OLSR lebih baik daripada *protocol* AODV. Hal tersebut karena pada *routing protocol* OLSR dalam mengirimkan paket data dari node sumber ke node tujuan dapat menyimpan informasi *routing* yang konsisten dan *up to date* pada setiap node sehingga dapat memberikan informasi rute dengan segera apabila diperlukan. Selain itu mekanisme MPR membuat proses pencarian rute menjadi efisien karena node MPR akan membuat paket OLSR yang diterima tidak akan langsung diteruskan ke node lain, tetapi hanya node yang dipilih sebagai MPR yang dapat menyampaikan paket kontrol yang diterima (Ainurrachman, Bhawiyuga, & Ichsan, 2017). Sehingga akan berdampak pada proses pencarian rute yang lebih cepat dari node sumber ke node tujuan.

Sedangkan pada *routing protocol* AODV dalam mengirimkan paket data dari node sumber ke node tujuan dengan melakukan *route discovery* informasi rute pada setiap node dan disimpan dalam tabel *routing* untuk setiap tujuan. Dalam proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk menemukan rute ke node tujuan. Apabila node menerima pesan RREQ yang memilki informasi rute ke node tujuan maka akan mengirimkan pesan RREP sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Namun hal tersebut akan berdampak pada proses pencarian rute yang berdampak pada waktu pengiriman paket yang lebih lama.

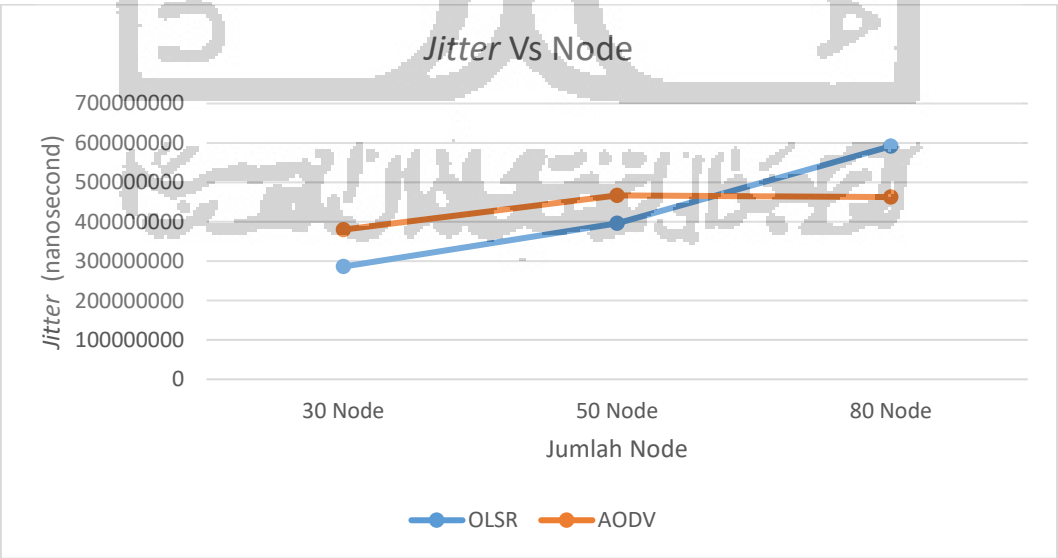
5. *Jitter*

Analisa dari hasil perhitungan rata-rata *Jitter* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario penambahan node dapat dilihat pada Tabel 4.5. Adapun satuan yang digunakan yaitu *nanoseconds*.

Tabel 4.5 Nilai Rata-Rata *Jitter* Pada Skenario Penambahan Node

Protocol	**30 Node**	**50 Node**	**80 Node**	**Rata-Rata**
OLSR	286106125	396157125	591630875	424631375
AODV	379874755	466767253	462124754	436255587

Dari data Tabel 4.5 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata *Jitter* dari skenario penambahan node yang terlihat pada Gambar 4.30.



Gambar 4.30 Grafik *Jitter* Vs Node

Berdasarkan grafik yang terlihat pada Gambar 4.30 menunjukkan bahwa nilai *Jitter* pada node 30 *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih *Jitter* 93768630 ns. Pada penambahan jumlah node 50, nilai *Jitter* kedua *protocol* mengalami peningkatan yaitu *protocol* OLSR sebesar 110051000 ns dari 286106125 ns menjadi 396157125 ns dan *protocol* AODV sebesar 86892498 ns dari 379874755 ns menjadi 466767253 ns dengan selisih keduanya sebesar 70610128 ns. Seiring pada kepadatan yang meningkat yaitu 80 node, *protocol* OLSR mengalami peningkatan nilai *Jitter* sebesar 195473750 ns dari 396157125 ns menjadi 591630875 ns. Sedangkan *protocol* AODV menurun cenderung stabil dengan nilai *Jitter* sebesar 4642499 ns dari 466767253 ns menjadi 462124754 ns dengan selisih keduanya sebesar 129506121 ns. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata *Jitter* 424631375 ns dan 436255587 ns dengan selisih mencapai 11624212 ns terhadap skenario penambahan jumlah node.

Pada Gambar 4.30 menunjukkan perbandingan nilai *Jitter* antara *protocol* OLSR dan *protocol* AODV dengan skenario penambahan jumlah node 30, 50, dan 80. Pada saat kedua *protocol* tersebut mengalami penambahan jumlah node dari 30 menjadi 50 node, kedua *protocol* mengalami penurunan nilai *Jitter*, hal itu karena pengaruh dari perubahan topologi yang menyebabkan banyaknya *hop* ke node tujuan menjadi berubah sehingga variasi *Delay* menjadi meningkat. Seiring bertambahnya node dari 50 menjadi 80 node terjadi sedikit penurunan nilai *Jitter* pada *protocol* AODV sedangkan *protocol* OLSR cenderung meningkat. Pada saat itu node sumber dalam mengirimkan paket data ke node tujuan dipengaruhi oleh perubahan topologi yang menyebabkan antrian paket data yang akan dikirimkan. Sedangkan *protocol* AODV dipengaruhi oleh proses *route discovery* .

Jika dilihat dari Gambar 4.30 menunjukkan grafik nilai *Jitter* dari *protocol* OLSR lebih baik daripada *protocol* AODV. Hal tersebut karena pada *routing protocol* OLSR dalam mengirimkan paket data dari node sumber ke node tujuan dapat menyimpan informasi *routing* yang konsisten dan *up to date* pada setiap node sehingga dapat memberikan informasi rute dengan segera apabila diperlukan. Selain itu mekanisme MPR membuat proses pencarian rute menjadi efisien karena node MPR akan membuat paket OLSR yang diterima tidak akan langsung diteruskan ke node lain, tetapi hanya node yang dipilih sebagai MPR yang dapat menyampaikan paket kontrol yang diterima (Ainurrachman, Bhawiyuga, & Ichsan, 2017). Sehingga akan berdampak pada proses pencarian rute yang lebih cepat dari node sumber ke node tujuan.

Sedangkan pada *routing protocol* AODV dalam mengirimkan paket data dari node sumber ke node tujuan dengan melakukan *route discovery* informasi rute pada setiap node dan disimpan dalam tabel routing untuk setiap tujuan. Dalam proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk menemukan rute ke node tujuan. Apabila node menerima pesan RREQ yang memilki informasi rute ke node tujuan maka akan mengirimkan pesan RREP sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Namun hal tersebut akan berdampak pada proses pencarian rute yang berdampak pada antrian paket data yang akan dikirimkan sehingga akan mempengaruhi banyaknya waktu yang dibutuhkan.

4.2.2 Hasil Simulasi Skenario Peningkatan Kecepatan

Pada simulasi skenario peningkatan kecepatan node dilakukan mulai dari 10 m/s, 15 m/s dan 20 m/s dengan area simulasi 1 km². Kecepatan yang digunakan untuk setiap node tetap dan jumlah node 35 dengan pola pergerakan secara acak. Kemudian hasil analisa performasi *routing protocol* OLSR dan AODV akan diukur berdasarkan parameter QoS yang dibagi dalam 4 bagian.

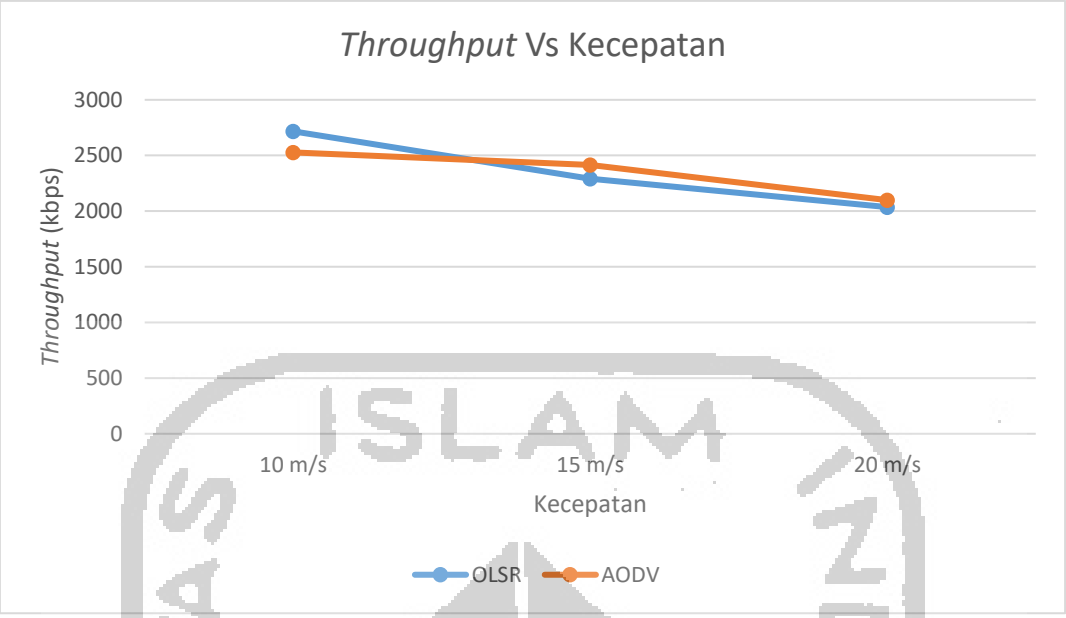
1. *Throughput*

Analisa dari hasil perhitungan rata-rata *Throughput* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario penambahan node dapat terlihat pada Tabel 4.6. Adapun satuan yang digunakan yaitu kbps *(kilo bit per second).*

Tabel 4.6 Nilai Rata-Rata *Throughput* Pada Skenario Peningkatan Kecepatan

Throughput	**10 m/s**	**15 m/s**	**20 m/s**	**Rata-Rata**
OLSR	2716,86	2291,38	2036,20	2348,15
AODV	2527,59	2415,38	2098,93	2347,30

Dari data Tabel 4.6 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata *Throughput* dari skenario peningkatan kecepatan node yang terlihat pada Gambar 4.31.



Gambar 4.31 Grafik *Throuhput* Vs Kecepatan

Berdasarkan grafik yang terlihat pada Gambar 4.31 menunjukkan bahwa nilai *Throughput* pada kecepatan node 10 m/s *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih 189,27 kbps. Namun pada peningkatan kecepatan node 15 m/s, *protocol* AODV mampu meningkatkan nilai *Throughput* sebesar 112,20 kbps dari 2527,59 kbps menjadi 2415,38 kbps daripada *protocol* OLSR menurun sebesar 425,48 kbps dari 2716,86 kbps menjadi 2291,38 kbps. Pada kecepatan node yang meningkat yaitu 20 m/s, *protocol* OLSR mengalami penurunan nilai *Throughput* sebesar 255,17 kbps dari 2291,38 kbps menjadi 2036,20 kbps dan *protocol* AODV sebesar 316,44 kbps dari 2415,38 kbps menjadi 2098,93 kbps. Dapat dilihat dari rata-rata *Throughput* kinerja *protocol* OLSR dan *protocol* AODV hampir sama. Kedua *protocol* dapat bekerja dengan maksimal dengan masing-masing nilai rata-rata *Throughput* yaitu 2348,15 kbps dan 2347,30 kbps terhadap skenario peningkatan kecepatan node.

Pada Gambar 4.31 menunjukkan perbandingan nilai *Throughput* antara *protocol* OLSR dan *protocol* AODV dengan skenario peningkatan kecepatan node 10 m/s, 15 m/s, dan 20 m/s. Pada saat kedua *protocol* tersebut terjadi peningkatan kecepatan node dari 10 m/s menjadi 15 m/s. Kedua *protocol* mengalami penurunan nilai *Throughput*. Hal itu karena pengaruh dari lebih cepatnya pergerakan node yang menyebabkan node sumber dalam mengirimkan paket data ke node tujuan semakin jauh jarak antar node sehingga akan mengalami putusnya rute. Seiring bertambahnya kecepatan node dari 15 m/s menjadi 20 m/s, kedua *protocol* mengalami

penurunan nilai *Throughput.* Pada saat itu jarak antar node satu dengan node yang lain menjadi semakin menjauh sehingga rute yang dilalui menjadi tidak tersedia.

Jika dilihat dari Gambar 4.31 menunjukkan grafik nilai *Throughput* dari *protocol* OLSR dan *protocol* AODV mempunyai kinerja yang cukup stabil. Hal tersebut karena pada kedua *routing protocol* dalam mengirimkan paket data dari node sumber ke node tujuan mampu memberikan kinerja yang baik. Pada *protocol* OLSR menggunakan mekanisme *Multi Point Relay* (MPR) untuk mengurangi pesan *broadcast* yang memiliki informasi rute yang sama dan disimpan dalam tabel *routing.*

Sedangkan pada *protocol* AODV dengan melakukan *route discovery* pada setiap node yang disimpan dalam tabel *routing* untuk setiap tujuan dan dengan melakukan *route discovery* pada setiap node yang disimpan dalam tabel *routing* untuk setiap tujuan sehingga keduanya dapat menurunkan penggunan *bandwidth.* Selain itu dengan jumlah node yang sama yaitu 35 node dan keberagaman tingkat kecepatan yang berbeda juga akan memberikan pengaruh dari hasil rata-rata nilai *Throughput* tersebut.

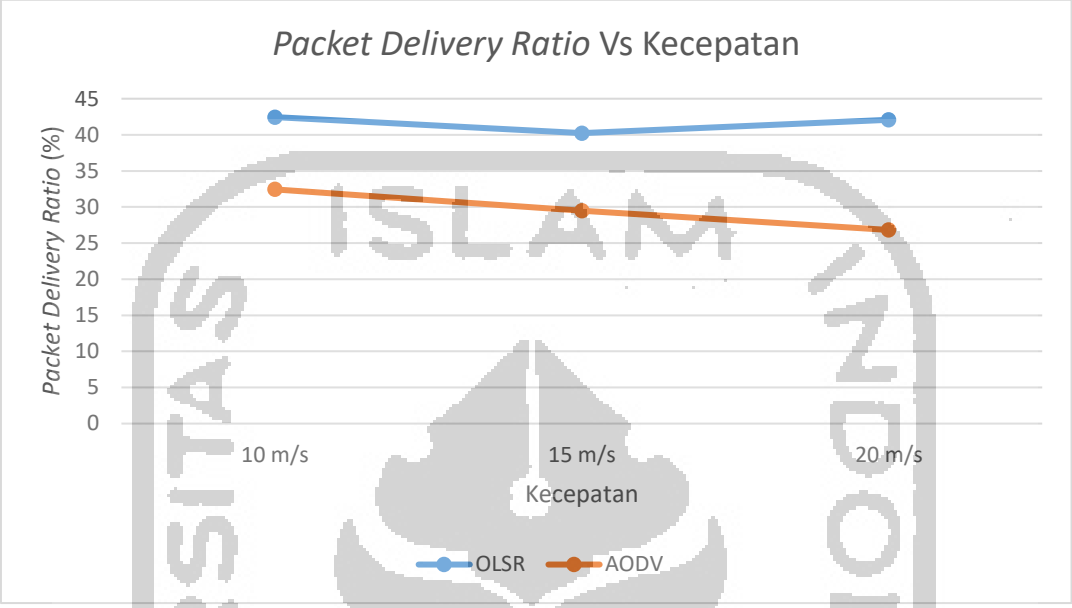
2. *Packet Delivery Ratio*

Analisa dari hasil perhitungan rata-rata *Packet Delivery Ratio* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario peningkatan kecepatan node dapat dilihat pada Tabel 4.7. Adapun satuan yang digunakan yaitu persentase.

Tabel 4.7 Nilai Rata-Rata PDR Pada Skenario Peningkatan Kecepatan

Protokol	**10 m/s**	**15 m/s**	**20 m/s**	**Rata-Rata**
OLSR	42,45	40,23	42,08	41,58
AODV	32,43	29,49	26,81	29,58

Dari data Tabel 4.7 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata PDR dari skenario peningkatan kecepatan node yang terlihat pada Gambar 4.32.



Gambar 4.32 Grafik *Packet Delivery Ratio* Vs Kecepatan

Berdasarkan grafik yang terlihat pada Gambar 4.32 menunjukkan bahwa persentase PDR pada peningkatan kecepatan node 10 m/s *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih pesentase 10,01 %. Pada peningkatan kecepatan node 15 m/s, persentase PDR *protocol* OLSR sedikit menurun sebesar 2,21 % dari 42,45 % menjadi 40,23 % sedangkan *protocol* AODV menurun sebesar 2,93 % dari 32,43 % menjadi 29,49 % dengan selisih keduanya mencapai 10,73 %. Pada kecepatan node yang meningkat yaitu 20 m/s, *protocol* OLSR sedikit meningkat dengan persentase PDR sebesar 1,84 % dari 40,23 % menjadi 42,08 % dan *protocol* AODV cenderung menurun sebesar 2,67 % dari 29,49 % menjadi 26,81 % dengan selisih keduanya mencapai 15,83 %. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata persentase PDR sebesar 41,58 % dan 29,58 % dengan selisih antara keduanya yaitu 12,00 % terhadap skenario peningkatan kecepatan node.

Pada Gambar 4.32 menunjukkan perbandingan nilai *Packet Delivery Ratio* antara *protocol* OLSR dan *protocol* AODV dengan skenario peningkatan kecepatan node 10 m/s, 15 m/s, dan 20 m/s. Pada saat kedua *protocol* tersebut terjadi peningkatan kecepatan node dari 10 m/s menjadi 15 m/s. Kedua *protocol* mengalami penurunan nilai *Packet Delivery Ratio*. Hal

itu karena pengaruh dari lebih cepat dari pergerakan node yang menyebabkan node sumber dalam mengirimkan paket data ke node tujuan semakin jauh jarak antar node sehingga mengalami putusnya rute atau rusak. Seiring bertambahnya kecepatan node dari 15 m/s menjadi 20 m/s, *protocol* OLSR mengalami sedikit peningkatan nilai *Packet Delivery Ratio.* Pada saat itu pergerakan node menjadi lebih cepat dan cukup stabil kinerja dari *protocol* OLSR, karena memiliki mekanisme MPR yang membuat proses pengiriman paket data menjadi lebih cepat. Berbeda halnya dengan *protocol* AODV yang mengalami penurunan nilai *Packet Delivery Ratio,* karena pengaruh dari proses *route discovery* sehingga persentase penerimaan paket data dari node sumber ke node tujuan semakin menurun.

Jika dilihat dari Gambar 4.32 menunjukkan grafik nilai *Packet Delivery Ratio* dari *protocol* OLSR lebih baik daripada *protocol* AODV. Hal tersebut karena pada *routing protocol* OLSR dalam mengirimkan paket data dari node sumber ke node tujuan dapat menyimpan informasi *routing* yang konsisten dan *up to date* pada setiap node sehingga dapat memberikan informasi rute dengan segera apabila diperlukan. Selain itu mekanisme MPR membuat proses pencarian rute menjadi efisien karena node MPR akan membuat paket OLSR yang diterima tidak akan langsung diteruskan ke node lain, tetapi hanya node yang dipilih sebagai MPR yang dapat menyampaikan paket kontrol yang diterima (Ainurrachman, Bhawiyuga, & Ichsan, 2017). Sehingga akan mempengaruhi tingkat keberhasilan paket yang dikirimkan.

Sedangkan pada *routing protocol* AODV dalam mengirimkan paket data dari node sumber ke node tujuan dibuat oleh node sumber. Apabila membutuhkan informasi *routing* ke node tujuan dengan melakukan *route discovery* yang diminta dengan membanjiri jaringan untuk mencari rute dan selesai ketika rute ditemukan (Training, 2016). *Route discovery* pada setiap node disimpan dalam tabel *routing* untuk setiap tujuan. Proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk mengetahui informasi rute ke node tujuan. Apabila node menerima pesan RREQ yang memiliki informasi rute ke node tujuan maka akan mengirimkan pesan RREP sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Hal tersebut dapat mempengaruhi dalam mengirimkan paket data ke node tujuan karena akan menunggu *route discovery* selesai dilakukan. Selain bertambahnya jumlah node akan membuat persentase PDR menurun karena akan membentuk banyak *hop* yang dilalui dan perubahan jalur topologi jaringan.

Selain itu tingkat kecepatan akan berpengaruh terhadap persentase PDR pada setiap kecepatan. Pada peningkatan kecepatan secara bertahap, *protocol* OLSR akan dengan cepat

mengirimkan paket ke node tujuan karena dapat menemukan informasi rute lebih cepat dan secara efisien sedangkan *protocol* AODV sedikit lebih rendah karena melakukan *discovery route* terlebih dahulu dalam mengirimkan paket data. Namun rendahnya pesentase PDR pada skenario peningkatan kecepatan node, karena dapat disebabkan beberapa hal, yaitu, tidak tersedianya rute karena node node sumber keluar dari jangkauan transmisi sinyal, banyaknya *hop* sehingga dapat merubah topologi dan *congestion* paket data dalam jaringan.

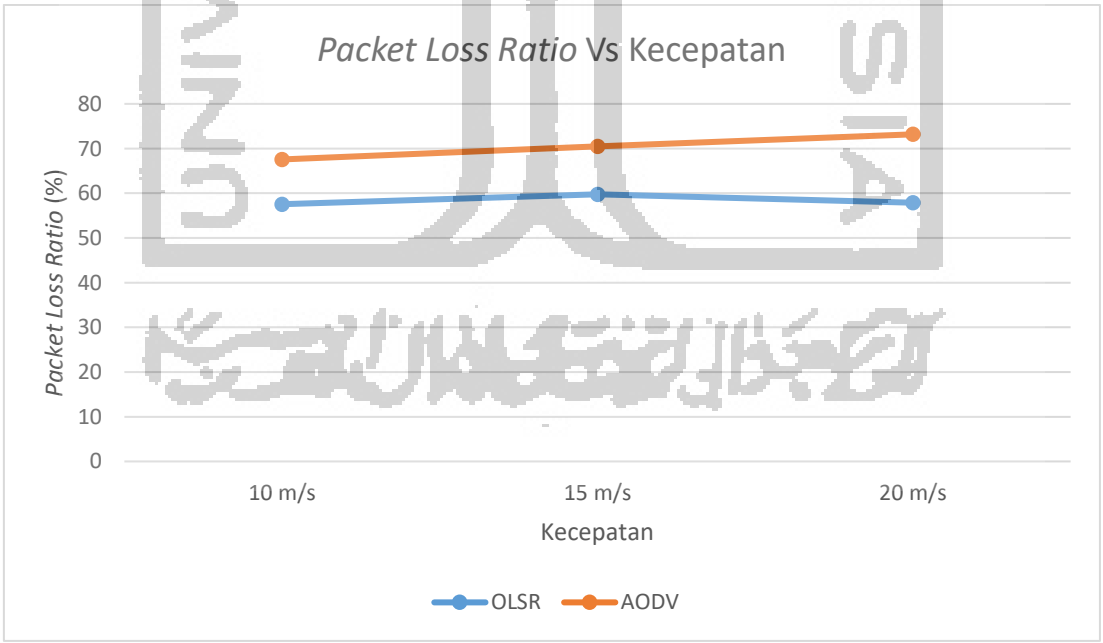
3. *Packet Loss Ratio*

Analisa dari hasil perhitungan rata-rata *Packet Loss Ratio* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario peningkatan kecepatan node dapat dilihat pada Tabel 4.8. Adapun satuan yang digunakan yaitu persentase.

Tabel 4.8 Nilai Rata-Rata PLR Pada Skenario Peningkatan Kecepatan

Protocol	**10 m/s**	**15 m/s**	**20 m/s**	**Rata-Rata**
OLSR	57,54	59,76	57,91	58,41
AODV	67,56	70,50	73,18	70,41

Dari data Tabel 4.8. dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata PLR dari skenario peningkatan kecepatan node yang terlihat pada Gambar 4.33.



Gambar 4.33 Grafik *Packet Loss Ratio* Vs Kecepatan

Berdasarkan grafik yang terlihat pada Gambar 4.33 menunjukkan bahwa persentase PLR pada peningkatan kecepatan node 10 m/s *protocol* OLSR lebih baik daripada *protocol* AODV dengan selisih persentase 10,01 %. Pada peningkatan kecepatan node 15 m/s, persentase PLR *protocol* OLSR sedikit meningkat sebesar 2,21 % dari 57,54 % menjadi 59,76 % sedangkan *protocol* AODV meningkat sebesar 2,93 % dari 67,56 % menjadi 70,50 %. Pada kecepatan node yang meningkat yaitu 20 m/s, *protocol* OLSR menurun persentase PLR sebesar 1,84 % dari 59,76 % menjadi 57,91 % daripada *protocol* AODV yang meningkat sebesar 2,67 % dari 70,50 % menjadi 73,18 %. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata persentase PLR sebesar 58,41 % dan 70,41 % terhadap skenario peningkatan kecepatan node.

Hasil dari *Packet Loss Ratio* (PLR) berbanding terbalik dengan hasil *Packet Delivery Ratio* (PDR) karena saling berkaitan satu sama lain, apabila nilai persentase dari PLR rendah atau menurun maka nilai persentase PDR tinggi atau meningkat dan sebaliknya. Karena pada nilai persentase PLR menggambarkan persentase paket hilang selama mengirimkan paket dan PDR menggambarkan persentase paket diterima oleh node tujuan.

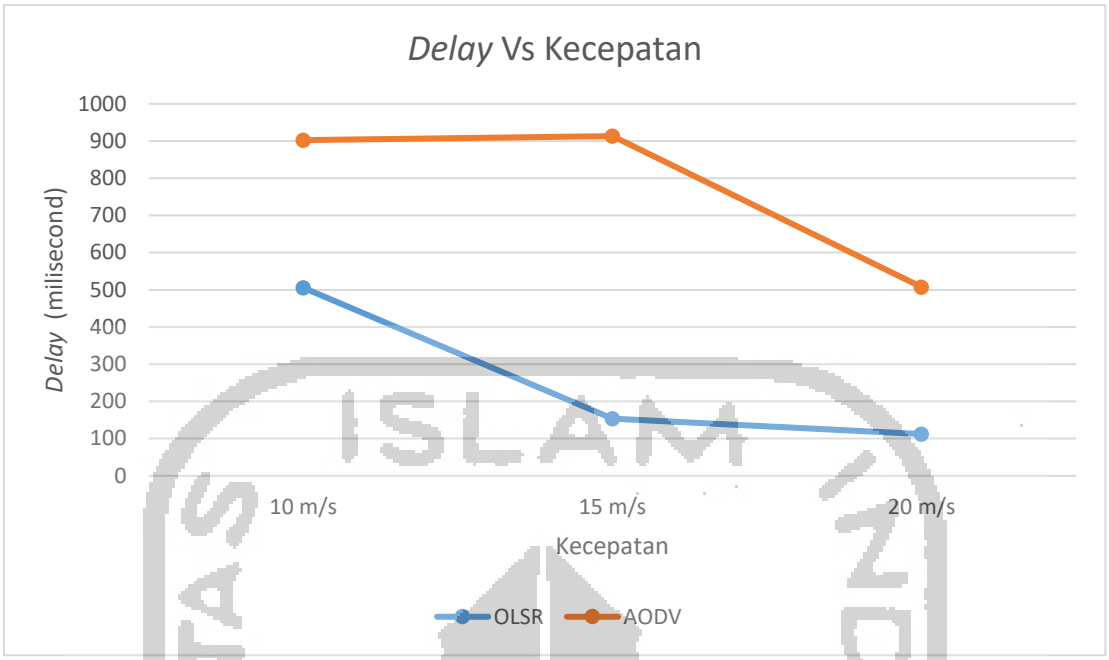
4. *Delay*

Analisa dari hasil perhitungan rata-rata *Delay* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario peningkatan kecepatan node dapat dilihat pada Tabel 4.9. Adapun satuan yang digunakan yaitu *milliseconds.*

Tabel 4.9 Nilai Rata-Rata *Delay* Pada Skenario Peningkatan Kecepatan

Protocol	**10 m/s**	**15 m/s**	**20 m/s**	**Rata Rata**
OLSR	505,54	153,31	112,19	257,02
AODV	902,09	913,19	507,46	774,25

Dari data Tabel 4.9 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata *Delay* dari skenario peningkatan kecepatan node yang terlihat pada Gambar 4.34.



Gambar 4.34 Grafik *Delay* Vs Kecepatan

Berdasarkan grafik yang terlihat pada Gambar 4.34 menunjukkan bahwa nilai *Delay* pada peningkatan kecepatan node 10 m/s *protocol* OLSR lebih baik daripada protokol AODV dengan selisih *Delay* 396,54 ms. Pada peningkatan kecepatan node 15 m/s, nilai *Delay protocol* OLSR mengalami penurunan sebesar 352,23 ms dari 505,54 ms menjadi 153,31 ms sedangkan *protocol* AODV sedikit meningkat sebesar 11,10 ms dari 902,09 ms menjadi 913,19 ms dengan selisih antara keduanya sebesar 759,87 ms. Pada kecepatan node yang meningkat yaitu 20 m/s, *protocol* AODV mampu menurunkan nilai *Delay* sebesar 405,72 ms dari 913,19 ms menjadi 507,46 ms dan *protocol* OLSR mengalami penurunan nilai *Delay* sebesar 41,11 ms dari 153,31 ms menjadi 112,19 ms dengan selisih antara keduanya mencapai 395,27 ms. Sehingga kinerja *protocol* OLSR lebih baik daripada *potocol* AODV yang masing-masing memiliki nilai rata-rata *Delay* 257,02 ms dan 774,25 ms dengan selisih antara keduanya sebesar 517,23 ms terhadap skenario peningkatan kecepatan node.

Pada Gambar 4.34 menunjukkan perbandingan nilai *Delay* antara *protocol* OLSR dan *protocol* AODV dengan skenario peningkatan kecepatan node 10 m/s, 15 m/s, dan 20 m/s. Pada saat kedua *protocol* tersebut terjadi peningkatan kecepatan node dari 10 m/s menjadi 15 m/s. *Protocol* OLSR mengalami penurunan nilai *Delay*. Hal itu karena pengaruh dari lebih cepatnya pergerakan node yang menyebabkan node sumber dalam mengirimkan paket data ke node tujuan menjadi semakin dekat jarak antar node sehingga waktu pengiriman data lebih cepat. Berbeda halnya dengan *protocol* AODV yang mengalami sedikit peningkatan nilai *Delay*,

karena pengaruh dari proses *route discovery* sehingga membutuhkan waktu yang lebih lama dalam mengirimkan paket data. Seiring bertambahnya kecepatan node dari 15 m/s menjadi 20 m/s, kedua *protocol* mengalami penurunan nilai *Delay*. Pada saat itu pergerakan node menjadi lebih cepat sehingga waktu proses pengiriman paket data dari node sumber ke node tujuan menjadi semakin cepat.

Tingkat kecepatan akan berpengaruh terhadap nilai *Delay* pada setiap kecepatan. Pada kecepatan 10 m/s, 15 m/s dan 20 m/s, *protocol* OLSR akan dengan cepat mengirimkan paket ke node tujuan karena dapat menemukan informasi rute dengan secara efisien karena pada OLSR terdapat mekanisme MPR. Mekanisme tersebut yang membuat proses pencarian rute menjadi efisien karena node MPR akan membuat paket OLSR yang diterima tidak akan langsung diteruskan ke node lain, tetapi hanya node yang dipilih sebagai MPR yang dapat menyampaikan paket kontrol yang diterima (Ainurrachman, Bhawiyuga, & Ichsan, 2017). Pada kecepatan tersebut semua node bergerak dengan laju stabil sehingga *protocol* OLSR akan dengan cepat mengirimkan paket data ke node tujuan dan menemukan informasi rute karena adaptif terhadap pengaruh dari perubahan topologi yang berubah dinamis.

Sedangkan pada *protocol* AODV dengan kecepatan 10 m/s, 15 m/s dan 20 m/s dalam mengirimkan paket data dari node sumber ke node tujuan dengan melakukan *route discovery* informasi rute pada setiap node dan disimpan dalam tabel *routing* untuk setiap tujuan. Dalam proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk menemukan rute ke node tujuan. Apabila node menerima pesan RREQ yang memiliki informasi rute ke node tujuan maka akan mengirimkan pesan RREP sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Pada kecepatan tersebut semua node bergerak dengan laju stabil yang akan berpengaruh pada tingkat kecepatan penerimaan paket data ke node tujuan sehingga membutuhkan waktu lebih lama akibat dari perubahan topologi.

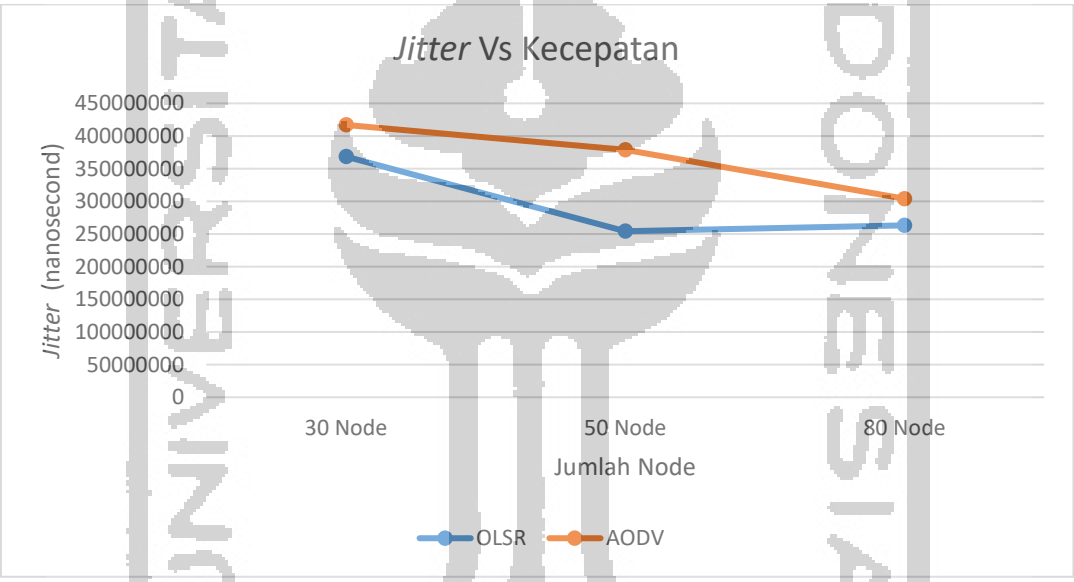
5. *Jitter*

Analisa dari hasil perhitungan rata-rata *Jitter* pada perbandingan performansi *routing protocol* OLSR dan AODV dengan skenario peningkatan kecepatan node dapat dilihat pada Tabel 4.10. Adapun satuan yang digunakan yaitu *nanoseconds.*

Tabel 4.10 Nilai Rata-Rata *Jitter* Pada Skenario Peningkatan Kecepatan Node

Protocol	**10 m/s**	**15 m/s**	**20 m/s**	**Rata-Rata**
OLSR	368397125	254189250	263102875	295229750
AODV	416896875	378840125	303973750	366570250

Dari data Tabel 4.10 dapat di gambarkan menjadi grafik yang merepresentasikan hasil perbandingan nilai rata-rata *Jitter* dari skenario peningkatan kecepatan node terlihat pada Gambar 4.35.



Gambar 4.35 Grafik *Jitter* Vs Node

Berdasarkan grafik yang terlihat pada Gambar 4.35 menunjukkan bahwa nilai *Jitter* pada peningkatan kecepatan node 10 m/s OLSR lebih baik daripada *protocol* AODV dengan selisih *Jitter* 48499750 ns. Pada peningkatan kecepatan node 15 m/s, nilai *Jitter* kedua *protocol* mengalami penurunan yaitu *protocol* OLSR sebesar 114207875 ns dari 368397125 ns menjadi 254189250 ns dan *protocol* AODV sebesar 38056750 ns dari 416896875 ns menjadi 378840125 ns dengan selisih keduanya sebesar 124650875 ns. Seiring pada bertambahnya kecepatan node yaitu 20 m/s, *protocol* OLSR mengalami peningkatan nilai *Jitter* sebesar 8913625 ns dari 254189250 ns menjadi 263102875 ns. Sedangkan *protocol* AODV cenderung menurun dengan nilai *Jitter* sebesar 74866375 ns dari 378840125 ns menjadi 303973750 ns

dengan selisih keduanya sebesar 40870875 ns. Sehingga kinerja *protocol* OLSR lebih baik daripada *protocol* AODV yang masing-masing memiliki nilai rata-rata *Jitter* 295229750 ns dan 366570250 ns dengan selisih mencapai 71340500 ns terhadap skenario penambahan jumlah node.

Pada Gambar 4.35 menunjukkan perbandingan nilai *Jitter* antara *protocol* OLSR dan *protocol* AODV dengan skenario peningkatan kecepatan node 10 m/s, 15 m/s, dan 20 m/s. Pada saat kedua *protocol* tersebut mengalami peningkatan kecepatan node dari 10 m/s menjadi 15 m/s, kedua *protocol* mengalami penurunan nilai *Jitter*, hal itu karena pengaruh dari perubahan topologi dan kecepatan node menjadi lebih cepat sehingga variasi *Delay* menjadi menurun. Seiring peningkatan kecepatn node dari 15 m/s menjadi 20 m/s node terjadi penurunan nilai *Jitter* pada *protocol* AODV sedangkan *protocol* OLSR cenderung meningkat. Pada saat itu node sumber dalam mengirimkan paket data ke node tujuan dipengaruhi oleh perubahan topologi dan kecepatan node sehingga menyebabkan antrian paket data yang akan dikirimkan. Sedangkan *protocol* AODV dipengaruhi oleh proses *route discovery*.

Jika dilihat dari Gambar 4.35 menunjukkan grafik nilai *Jitter* dari *protocol* OLSR lebih baik daripada *protocol* AODV. Hal tersebut karena pada *routing protocol* OLSR dalam mengirimkan paket data dari node sumber ke node tujuan dapat menyimpan informasi *routing* yang konsisten dan *up to date* pada setiap node sehingga dapat memberikan informasi rute dengan segera apabila diperlukan. Selain itu mekanisme MPR membuat proses pencarian rute menjadi efisien karena node MPR akan membuat paket OLSR yang diterima tidak akan langsung diteruskan ke node lain, tetapi hanya node yang dipilih sebagai MPR yang dapat menyampaikan paket kontrol yang diterima (Ainurrachman, Bhawiyuga, & Ichsan, 2017). Sehingga akan berdampak pada proses pencarian rute yang lebih cepat dari node sumber ke node tujuan.

Sedangkan pada *routing protocol* AODV dalam mengirimkan paket data dari node sumber ke node tujuan dengan melakukan *route discovery* informasi rute pada setiap node dan disimpan dalam tabel routing untuk setiap tujuan. Dalam proses *route discovery* dengan meneruskan pesan *broadcast* menuju node tetangga untuk menemukan rute ke node tujuan. Apabila node menerima pesan RREQ yang memilki informasi rute ke node tujuan maka akan mengirimkan pesan RREP sedangkan apabila tidak memiliki informasi rute tujuan maka akan mengirimkan *broadcast* ulang RREQ ke node tetangganya (Rezkinanda & Anggoro, 2016). Namun hal tersebut akan berdampak pada proses pencarian rute yang berdampak pada antrian

paket data yang akan dikirimkan sehingga akan mempengaruhi banyaknya waktu yang dibutuhkan.

4.3 Analisa Hasil Simulasi

Pada analisa hasil simulasi ini akan menganalisa hasil dari performansi kinerja *routing protocol* OLSR dan *routing protocol* AODV berdasarkan parameter QoS, seperti *Throughput, Packet Deivey Ratio* (PDR), *Packet Loss Ratio* (PLR), *Delay* dan *Jitter*.

Pada skenario penambahan jumlah node, penelitian ini menggunakan node berjumlah 30, 50 dan 80 dengan luas area 1 km² yang ditempatkan secara *random*. Dalam proses komunikasi antara node sumber ke node tujuan dalam mengirimkan paket data dilakukan secara *random* dengan node pengirim dan penerima berjumlah 8 pasang yang dilakukan *one hop* ataupun *multi-hop*. Untuk pergerakan semua node bersifat *random* dengan kecepatan antara 0 m/s sampai 20 m/s. Node pengirim dan node penerima tersebut berlaku untuk semua skenario penambahan jumlah node. Adapun hasil simulasi yang telah dilakukan pada skenario penambahan jumlah node dari 30 node, 50 node dan 80 node dengan kecepatan 0-20 m/s. Dapat diketahui bahwa performa *routing protocol* OLSR lebih baik dibandingkan dengan *routing protocol* AODV dari nilai rata-rata *Throughput, Packet Delivery Ratio, Packet Loss Rati, Delay dan Jitter*.

Sedangkan pada skenario peningkatan kecepatan node, penelitian ini menggunakan node berjumlah 30 dengan luas area 1 km² yang ditempatkan secara *random* untuk semua kecepatan. Dalam proses komunikasi antara node sumber ke node tujuan dalam mengirimkan paket data dilakukan secara *random* dengan node pengirim dan penerima berjumlah 8 pasang yang dilakukan *one hop* ataupun *multi-hop*. Untuk pergerakan node bersifat tetap dengan kecepatan 10 m/s, 15 m/s dan 20 m/s. Node pengirim dan node penerima tersebut berlaku untuk semua skenario peningkatan kecepatan node. Adapun hasil simulasi yang telah dilakukan pada skenario peningkatan kecepatan node dari 10 m/s, 15 m/s dan 20 m/s dengan kecepatan tetap dan jumlah node 35 yang bergerak secara konstant. Dapat diketahui bahwa performa *routing protocol* OLSR secara keseluruhan dari nilai rata-rata *Throughput, Packet Delivery Ratio, Packet Loss Ratio*, *Delay dan Jitter* lebih baik dan cenderung stabil dibandingkan dengan *routing protocol* AODV.

Berdasarkan hasil performansi *routing protocol* OLSR dan AODV yang diukur menggunakan parameter QoS, seperti *Throughput, Packet Deivey Ratio* (PDR), *Packet Loss Ratio* (PLR), *Delay* dan *Jitter* dengan skenario penambahan jumlah node dan peningkatan

kecepatan node, diketahui bahwa *routing protocol* OLSR dapat digunakan pada tingkat kepadatan jaringan menengah dan dengan kecepatan sedang. Hal tersebut karena *protocol* OLSR memiliki mekanisme MPR yang dapat membuat proses pencarian rute menjadi efisien sehingga proses pengiriman paket data menjadi lebih cepat. Sedangkan pada *routing protocol* AODV dapat digunakan pada tingkat kepadatan berskala kecil dan dengan kecepatan rendah.

Adapun dalam penelitian ini terdapat beberapa kelebihan dan kekurangan dalam melakukan simulasi jaringan VANET menggunakan simulator NS-3. Kelebihan dan kekurangan dalam penelitian ini diharapkan menjadi bahan rujukan dan pertimbangan sehingga dapat mengembangkan jaringan VANET lebih lanjut dan pada kondisi yang sebenarnya. Oleh karena itu, di bawah ini merupakan kelebihan dan kekurangan simulasi jaringan VANET yang telah dihasilkan, antara lain:

a. Kelebihan
   - Pada simulasi ini membandingkan *protocol topology based* dari jenis *protocol proactive* yaitu OLSR dengan *protocol reactive* yaitu AODV.
   - Pada simulasi ini mengukur kinerja performansi dari *routing protocol* OLSR dan AODV menggunakan parameter *Quality of Service* (QoS), seperti *Throughput, Packet Delivery Ratio, Packet Loss Rate, Delay* dan *Jitter*.
   - Pada saat simulasi berlangsung terdapat tampilan *capture* dari proses pengiriman paket data dari node sumber ke node tujuan dan juga terdapat tampilan proses visualisasi simulasi jaringan yang dijalankan secara langsung.
   - Pada simulasi ini data hasil QoS setelah simulasi selesai dilakukan dapat ditampilkan secara detail.

b. Kekurangan
   - Diperlukan kajian lebih lanjut mengenai pengembangan *routing protocol* OLSR atau *routing protocol* lain yang lebih adaptif terhadap lingkungan VANET sehingga lebih optimal dari segi QoS di antaranya, *Throughput*, *Packet Delivery Ratio, Delay* dan *Jitter*. Karena pada penelitian ini nilai *Throughput, Packet Delivery Ratio, Delay* dan *Jitter* pada skenario penambahan jumlah node masih rendah. Hal tersebut akan berpengaruh terhadap pengiriman paket data dari node sumber ke node tujuan menjadi hilang atau rusak sehingga pakat data yang dikirimakan tidak sampai. Selain itu juga tidak tersedianya rute karena node node sumber keluar dari jangkauan transmisi sinyal, pergerakan node yang acak,

perubahan topologi, perbedaan parameter dan *congestion* paket data dalam jaringan.

- Pada simulasi ini belum menghitung parameter QoS lain, seperti *Routing Overhead* dan lainnya. Karena pada simulasi ini memiliki keterbatasan data yang dihasilkan untuk menghitung parameter tersebut. Parameter tersebut penting karena *Routing Overhead* atau *Routing Load* akan menghitung jumlah besarnya paket kontrol informasi yang dihasilkan oleh node sumber untuk mengirimkan paket data ke node tujuan
- Pada simulasi ini hanya memodelkan jaringan VANET secara *Vehicle to Vehicle* (V2V) dengan suatu wilayah berbentuk grid sehingga dapat melakukan simulasi dengan memodelkan rancangan peta suatu wilayah dengan terhubungnya infrastruktur jalan atau *Road Side Unit* (RSU) terhadap beberapa kendaraan sehingga dapat diukur tingkat QoS jaringan VANET pada kondisi yang mendekati sebenarnya. Karena pada simulasi ini model rancangan simulasi masih sederhana sehingga data hasil QoS yang dihasilkan kurang akurat apabila diterapkan pada kondisi yang sebenarnya.

4.4 Konsep Implementasi VANET

Konsep implemetasi dari hasil simulasi dengan skenario penambahan jumlah node dan peningkatan kecepatan node dapat digunakan pada jaringan VANET secara *Vehicle to Vehicle* (V2V) untuk aplikasi real time, seperti layanan pesan keselamatan pada kendaraan, layanan pesan darurat kepada kendaraan lain, *traffic signal* dan lain-lain. Sebagai contoh, salah satunya pada kendaraan yang memiliki prioritas penggunaan jalan tinggi yaitu ambulan atau kendaraan pemadadam kebakaran, ketika terjadi suatu bencana kebakaran pada suatu wilayah. Pada kendaraan tersebut dapat melakukan *broadcast* pesan darurat kepada kendaraan lainnya yang berada pada jalan menuju tempat bencana kebakaran tujuan untuk memberikan prioritas penggunaan jalan agar dapat cepat sampai. Hal tersebut sangat penting karena akan mempercepat proses pemadaman dan relokasi penduduk sekitar. Implementasi VANET pada kendaraan tersebut penting, mengingat teknologi yang digunakan pada kendaraan pemadam kebakaran atau ambulan menggunakan teknologi suara sirine yang terbatasnya pada jarak yang disiarkan. Sedangkan apabila menggunakan teknologi VANET dapat mempercepat pesan penyiaran darurat karena ditransmisikan melalui jaringan *wireless* secara *one-hop* atau *multi-hop* dengan memanfaatkan pada kendaraan sekitar atau *Road Side Unit* (RSU) untuk

menyiarkan pesan darurat tersebut. Pada kondisi tersebut, pemanfaatan *routing protocol* OLSR dapat digunakan karena memiliki performa yang optimal pada tingkat kepadatan yang meningkat dan kecepatan yang cenderung sedang. Hal tersebut karena pengaruh dari mekanisme MPR yang dapat membuat proses pencarian rute menjadi efisien sehingga proses pengiriman paket data menjadi lebih cepat.

Sedangkan pada konsep implementasi pemanfaatan *routing protocol* AODV pada jaringan VANET pada saat terjadi bencana (tsunami, gunung meletus dan lain-lain) untuk relokasi penduduk sekitar. Diasumsikan pada kendaraan-kendaraan penduduk telah menggunakan jaringan VANET sehingga akan mempercepat relokasi. Dengan membentuk jaringan VANET secara kluster atau jumlah kecil untuk menuju relokasi ke tempat yang lebih aman. Pada kluster tersebut terdapat kendaraan pemimpin atau *leader* yang bertujuan untuk memberitahu jalur relokasi dan memimpin kendaraan yang disekitarnya.