Analysis on Routing Algorithms in Various Traffic Scenarios (VANET)

Da Teng, Jiacheng Zhang

Introduction

Objective

- Investigate the performance of various routing algorithms in different scenarios
- Assess the suitability of algorithms for specific environment types

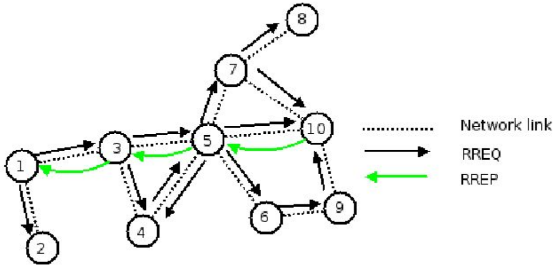
Scenarios for Evaluation

- Campus low pedestrian traffic
- Campus high pedestrian traffic
- Countryside low vehicular traffic
- Urban high vehicular traffic

https://stl.tech/blog/naas-now-later-how-network-as-a-service-benefits-your-business/

AODV (Ad-hoc On-demand Distance Vector)

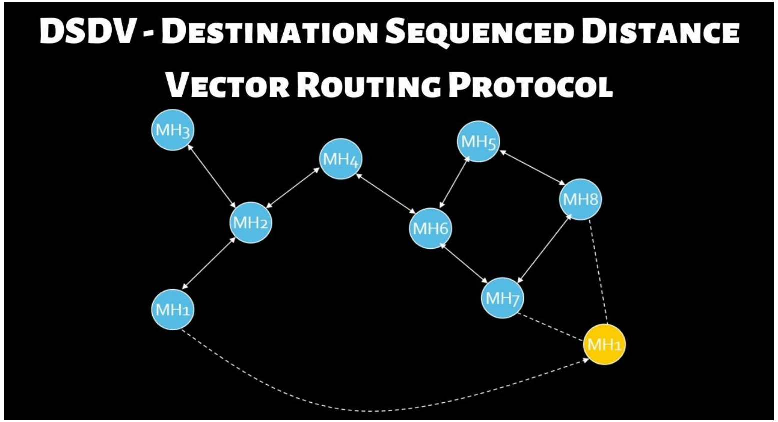
- Reactive routing protocol.
- Routes are established on-demand when needed.
- Nodes maintain routes to destinations through route discovery and route maintenance processes.
- Utilizes sequence numbers to ensure the freshness of rc information and prevent routing loops.
- Well-suited for ad-hoc networks with mobile nodes and unpredictable network topology changes.



https://www.researchgate.net/figure/AODV-routing-protocol-showing-the-route-discovery-process_fig2_220134116

DSDV (Destination-Sequenced Distance Vector)

- Proactive routing protocol.
- Each node maintains a routing table containing the next hop for each destination along with a sequence number.

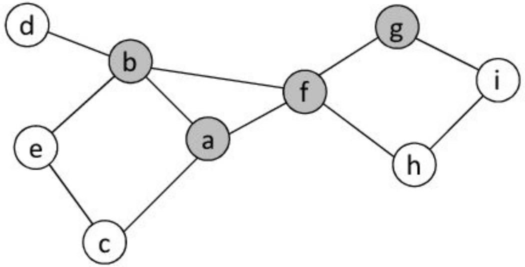


https://i.ytimg.com/vi/vJ8SD5ulGps/maxresdefault.jpg

- Periodic updates are sent to neighboring nodes to advertise changes in routes.
- Uses sequence numbers to ensure the freshness of routing information and avoid routing loops.
- Suitable for relatively stable networks where periodic updates are feasible and overhead is acceptable.

OLSR (Optimized Link State Routing)

- Proactive routing protocol
- Nodes maintain topology information through periodic exchange of link state information.
- Each node calculates shortest paths to all destinations using the information collected.
- Utilizes multipoint relays (MPRs) to reduce overhead by selectively flooding control messages.
- Suitable for highly dynamic networks with frequent topology changes.



MPRs	Selector Set
a	b, c, f
b	a, e, f, d
f	a, b, h, g
g	f, i

https://www.researchgate.net/figure/AODV-routing-protocol-showing-the-route-discovery-process_fig2_220134116

Predictions

- OLSR: Efficient in stable, low-traffic environments; faces challenges in high-traffic scenarios; proactive nature suits predictable and low-traffic settings.
- AODV: Adapts well to dynamic changes, particularly in high-traffic scenarios; maintains efficiency across varying traffic levels; provides responsive on-demand route establishment.
- DSDV: Adequate in stable, low-traffic conditions; struggles in high-traffic scenarios due to periodic updates; performs decently in low-traffic scenarios but may encounter routing challenges over time.

Approach

- OSMWebwizard generates scenario
- Convert into ns2mobility.tcl files
- Filter the simulation to preserve only the period of interest
- Utilizes NS-2 mobility trace files to dictate node movement
- Sets up packet sending and receiving functionalities on nodes
- Store all packet flow information into .flowmon files
- Parse .flowmon files to extract useful information
- Evaluate performance with 3 metrics:

  Packet Loss Ratio, Delay, Number of Hops

Georgia Tech

SUMO Simulation

Simulation Flow

- Initialization of node properties and wireless channel characteristics.
- Application of mobility model from an external NS-2 trace file.
- Dynamic assignment of IP addresses based on the number of nodes.
- Configuration of end-to-end communication between nodes using the specified routing protocol.
- Collection and output of simulation results through logs and FlowMonitor reports.
- Utilize ns3 built-in parsing script for .flowmon file and our own customized scripts to extract useful information and make data plots.

Wi-Fi PHY Layer/Channel Setup

```
WifiHelper wifi;
wifi.SetStandard(WIFI_STANDARD_80211b);

YansWifiPhyHelper wifiPhy;

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel(wifiChannel.Create());

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

//Using the bulit-in ns-2 mobility helper
Ns2MobilityHelper sumo_trace (FileNamePrefix + location +".tcl");
sumo_trace.Install(); //install ns-2 mobility in all nodes
```

IP Address Dynamic Allocation

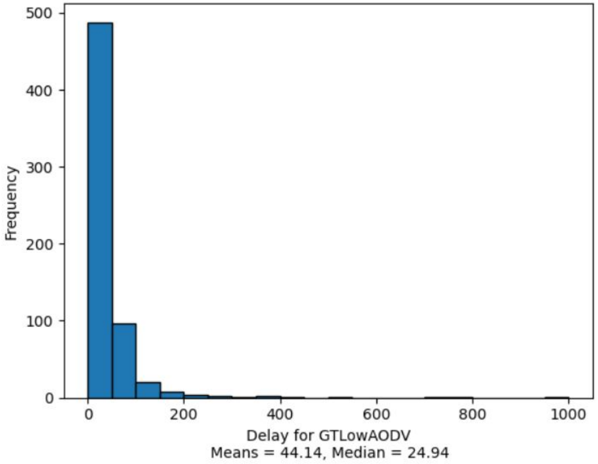
```
Ipv4AddressHelper address;
if (nnodes <= 254) // up to 254 usable IP addresses
{
    address.SetBase("10.1.1.0", "255.255.255.0");
}
else if (nnodes <= 510) // 255 to 510 usable IP addresses
{
    address.SetBase("10.1.0.0", "255.255.254.0");
}
else if (nnodes <= 1022) // 511 to 1022 usable IP addresses
{
    address.SetBase("10.1.0.0", "255.255.252.0");
}
else if (nnodes <= 2046) // 1023 to 2046 usable IP addresses
{
    address.SetBase("10.1.0.0", "255.255.248.0");
}
```

Packet Sending/Receiving Mechanism

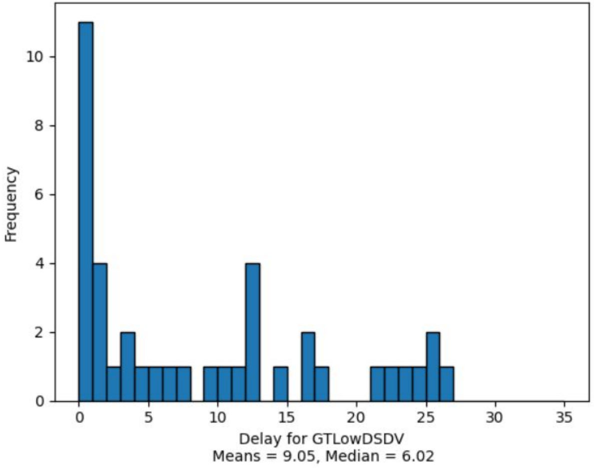
```
// Set up applications to send packets from each node to the previous node, starting from node 2
for (uint32_t i = 1; i < nnodes; ++i) {
    uint32_t senderIndex = i;
    uint32_t receiverIndex = i - 1;

    Ptr<Socket> sink = SetupPacketReceive(adhocInterfaces.GetAddress(receiverIndex), adhocNodes.Get(receiverIndex));
    Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable>();

    OnOffHelper onoff("ns3::UdpSocketFactory", Address(InetSocketAddress(adhocInterfaces.GetAddress(receiverIndex), port)));
    onoff.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1.0]"));
    onoff.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0.0]"));
    ApplicationContainer temp = onoff.Install(adhocNodes.Get(senderIndex));
    temp.Start(Seconds(var->GetValue(sim_time *4.0 / 5,sim_time *4.0 /5 + 3.0)));
    temp.Stop(Seconds(sim_time));
}
```

Results (GT Low Pedestrian Traffic)

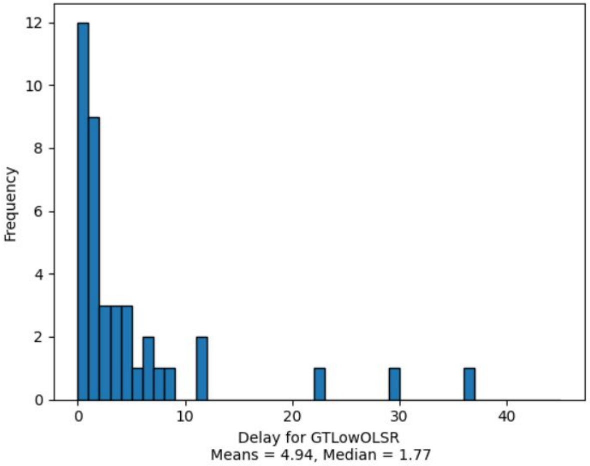
AODV



DSDV



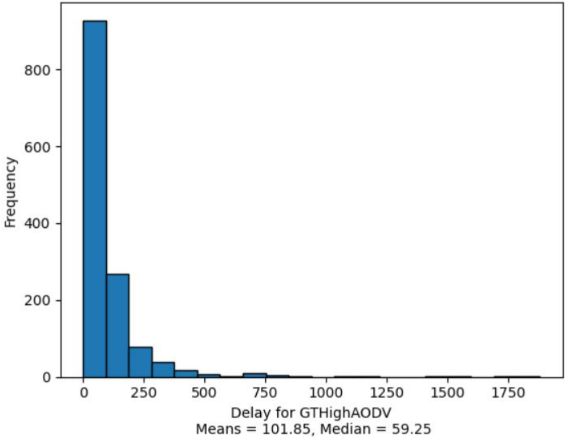
OLSR



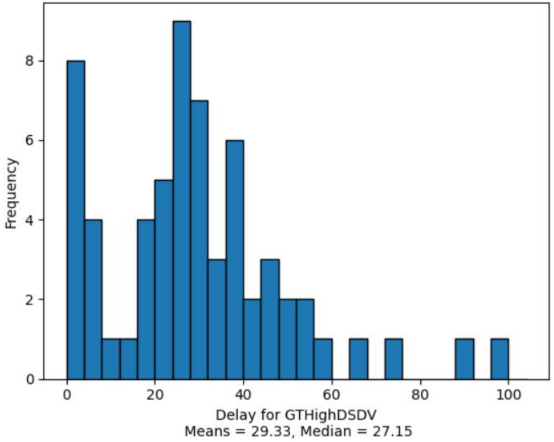
Georgia Tech

Results (GT High Pedestrian Traffic)

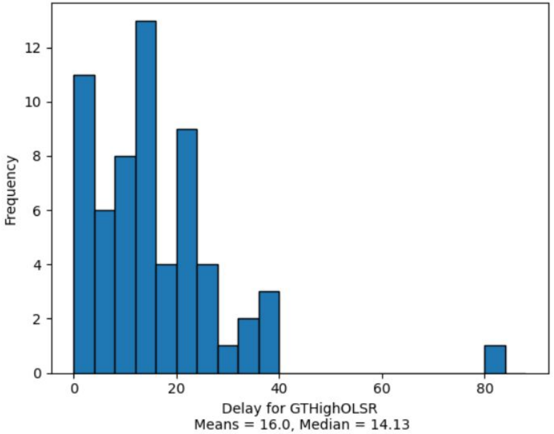
AODV



DSDV



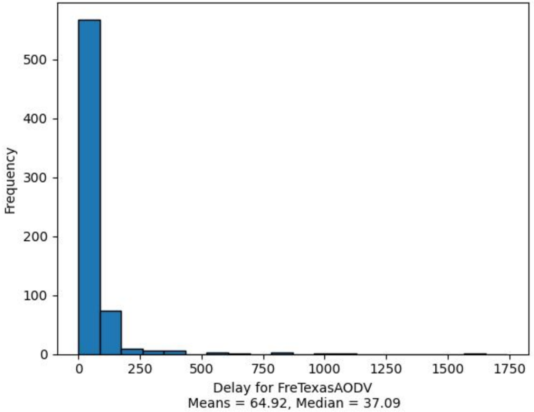
OLSR



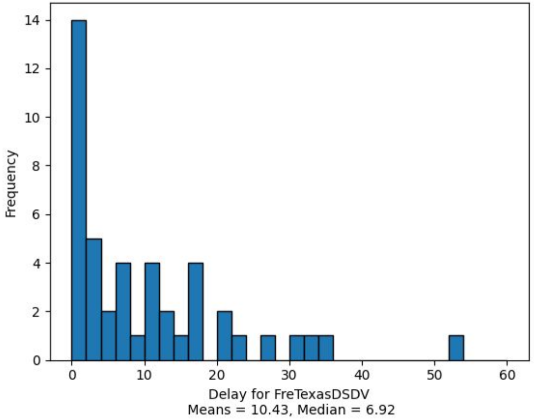
Georgia Tech

Results (Low Vehicular Traffic)

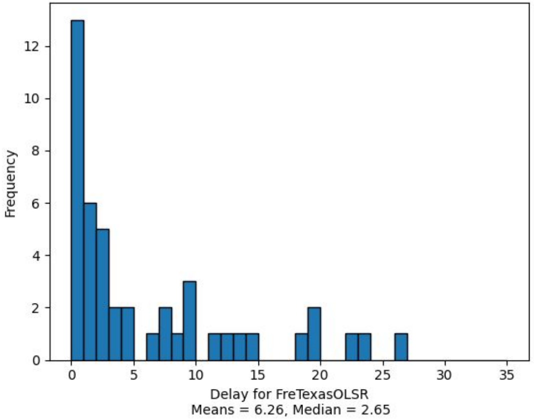
AODV



DSDV



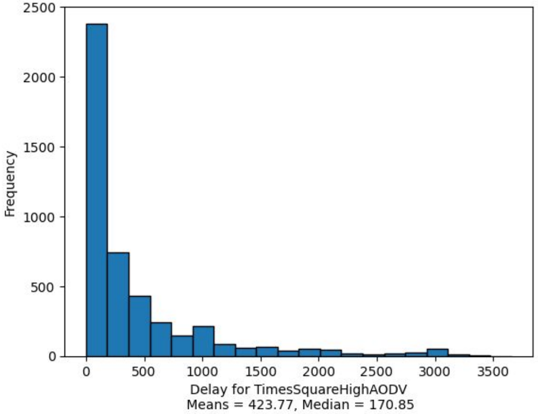
OLSR



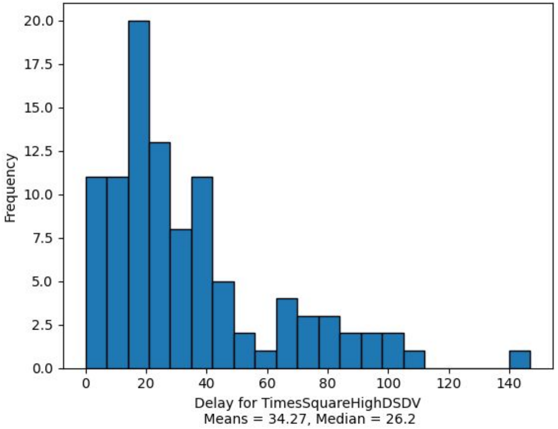
Georgia Tech

Results (High Vehicular Traffic)

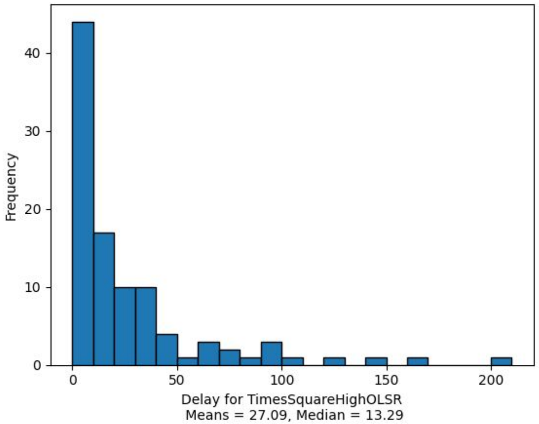
AODV



DSDV



OLSR



Conclusion

- Within the same traffic scenarios, OLSR has a better performance since it has a shorter delay compared to AODV and DSDV.
- However, other performance metrics need to be taken into account
- In terms of packet loss ratio and number of hops, the same result does not hold
- Further detailed analysis in the report.

Georgia Tech

Future Work

- Other more sophisticated evaluation metrics such as energy consumption, fairness, scalability, etc
- Test on highway vehicles
- More complex sending/receiving mechanism
- DSR algorithm