

**TUGAS AKHIR – IF184802**

**IMPLEMENTASI K-MEANS CLUSTERING DAN MODIFIKASI ADHOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL PADA MOBILE ADHOC NETWORK UNTUK MENINGKATKAN PERFORMA PENGIRIMAN ANTAR NODE**

**FAHRIZAL NAUFAL AHMAD**

**NRP 05111640000135**

**Dosen Pembimbing I**

**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

**Dosen Pembimbing II**

**Ir.F.X. Arunanto M.Sc.**

**Departemen Teknik Informatika**

**Fakultas Teknologi Elektro dan Informatika Cerdas**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2020**

***(Halaman ini sengaja dikosongkan)***



**TUGAS AKHIR – IF184802**

**IMPLEMENTASI K-MEANS CLUSTERING DAN MODIFIKASI ADHOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL PADA MOBILE ADHOC NETWORK UNTUK MENINGKATKAN PERFORMA PENGIRIMAN ANTAR NODE**

**FAHRIZAL NAUFAL AHMAD**

**NRP 05111640000135**

**Dosen Pembimbing I**

**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

**Dosen Pembimbing II**

**Ir.F.X. Arunanto M.Sc.**

**Departemen Teknik Informatika**

**Fakultas Teknologi Elektro dan Informatika Cerdas**

**Institut Teknologi Sepuluh Nopember**

**Surabaya 2020**

***(Halaman ini sengaja dikosongkan)***



**UNDERGRADUATE THESIS – IF184802**

**IMPLEMENTATION OF K-MEANS CLUSTERING AND BACKUP ROUTING ALGORITHM IN ADHOC ON-DEMAND DISTANCE VECTOR TO ENHANCE MOBILE NODE PERFORMANCE ON WIRELESS ADHOC NETWORK**

**FAHRIZAL NAUFAL AHMAD**

**NRP 05111640000135**

**First Advisor**

**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

**Second Advisor**

**Ir.F.X. Arunanto M.Sc.**

**Department of Informatics Engineering**

**Faculty of Electrical Technology and Intelligent Informatics**

**Sepuluh Nopember Institute of Technology**

**Surabaya 2020**

***(Halaman ini sengaja dikosongkan)***

**LEMBAR PENGESAHAN**

**IMPLEMENTASI K-MEANS CLUSTERING DAN MODIFIKASI ADHOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL PADA MOBILE ADHOC NETWORK UNTUK MENINGKATKAN PERFORMA PENGIRIMAN ANTAR NODE**

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

pada

Bidang Studi Arsitektur dan Jaringan Komputer

Program Studi S-1 Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Oleh:

**FAHRIZAL NAUFAL AHMAD**

**NRP: 05111640000135**

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. .......................

(NIP. 198410162008121002) (Pembimbing 1)

2. Ir.F.X. Arunanto M.Sc. ........................

(NIP. 195701011983031004) (Pembimbing 2)

**SURABAYA**

**JUNI, 2020**

***(Halaman ini sengaja dikosongkan)***

**IMPLEMENTASI K-MEANS CLUSTERING DAN MODIFIKASI ADHOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL PADA MOBILE ADHOC NETWORK UNTUK MENINGKATKAN PERFORMA PENGIRIMAN ANTAR NODE**

|  |  |
| --- | --- |
| **Nama Mahasiswa** | **: Fahrizal Naufal Ahmad** |
| **NRP** | **: 05111640000135** |
| **Departemen** | **: Teknik Informatika** |
| **Dosen Pembimbing 1** | **: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.** |
| **Dosen Pembimbing 2** | **: Ir.F.X. Arunanto M.Sc.** |
|  |  |

# Abstrak

*Adhoc On-Demand Distance Vector*(AODV) merupakan salah satu *routing protocol* yang dapat diaplikasikan ke pada *Mobile Adhoc Network* (MANET). AODV sendiri memiliki dua fase, yaitu *route discovery* dan *route* maintenance. *Route Discovery* sendiri memiliki dua fase yaitu *Route Request* (RREQ) dan *Route Reply* (RREP)

Pada AODV, *route* yang dipilih adalah *route* dengan jumlah hop terkecil tanpa memperhatikan faktor-faktor esensial lainnya yang dapat memengaruhi dalam pemilihan suatu *route*. Ada beberapa faktor lain yang bisa mempengaruhi pemilihan *route* selain jumlah hop antara lain seperti posisi, energi, kepadatan, dan kekuatan sinyal. Penggunaan faktor-faktor tambahan tersebut akan meningkatkan akurasi didalam pemilihan *route*.

Pada Tugas Akhir ini, Penulis mengusulkan sebuah algoritma bernama *Backup Routing* untuk diimplementasikan di dalam AODV. Untuk lebih meningkatkan performa, Penulis juga mengusulkan sebuah metode *clustering*, yaitu *K-Means Clustering.* Dua metode tersebut akan menghitung seluruh rute yang tersedia lalu melakukan *clustering* sebelum mengirimkan sebuah paket.

Pada akhir pengujian, dua metode tersebut yang telah diimplementasikan akan disimulasikan. Setelah disimulasikan, terdapat data yang menunjukkan bahwa adanya peningkatan performa dari AODV yang telah dimodifikasi.

***Kata kunci: MANET, AODV, Backup Routing, K-Means, Clustering***

**IMPLEMENTATION OF K-MEANS CLUSTERING AND BACKUP ROUTING ALGORITHM IN ADHOC ON-DEMAND DISTANCE VECTOR TO ENHANCE MOBILE NODE PERFORMANCE ON WIRELESS ADHOC NETWORK**

|  |  |
| --- | --- |
| **Student Name** | **: Fahrizal Naufal Ahmad** |
| **Student ID** | **: 05111640000135** |
| **Department** | **: Teknik Informatika** |
| **First Advisor** | **: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.** |
| **Second Advisor** | **: Ir.F.X. Arunanto M.Sc.** |
|  |  |

# *Abstract*

*Adhoc On-Demand Distance Vector (AODV) is a routing protocol that can be applied to the Mobile Adhoc Network (MANET). AODV itself has two phases, namely route discovery and route maintenance. Route Discovery itself has two phases namely Route Request (RREQ) and Route Reply (RREP)*

*In AODV, the selected route is the route with the smallest number of hops without looking to other essential factors that can influence the selection of a route. There are several factors that can influence route selection besides the number of hops, such as position, energy, density, and signal strength. Using these additional factors will improve accuracy in route selection.*

*In this Undergraduate Thesis, the author proposes an algorithm called Backup Routing to be implemented in AODV. To further improve performance, the author also proposes a clustering method, namely K-Means Clustering. The two methods will calculate all available routes and then cluster before sending a packet.*

*At the end of the test, the two methods that have been implemented will be simulated. After being simulated, there are data that show an increase in the performance of the modified AODV.*

***Keyword: MANET, AODV, Backup Routing, K-Means, Clustering***

**KATA PENGANTAR**

Puji syukur kepada Allah SWT atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul ***“IMPLEMENTASI K-MEANS CLUSTERING DAN ALGORITMA BACKUP ROUTING PADA AD-HOC ON-DEMAND DISTANCE VECTOR UNTUK MENINGKATKAN PERFORMA MOBILE NODE PADA MOBILE AD-HOC NETWORK*”.**

Harapan penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas semua rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Keluarga penulis (Bapak Muhammad Thobii, Ibu Maria Magdalena Sri Haryani, Maulida Rahma Britania, dan Risyad Athaya Muhammad) yang selalu memberikan dukungan baik berupa doa, moral, dan materi kepada penulis, sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. dan Bapak Ir.F.X. Arunanto M.Sc. selaku dosen pembimbing, atas arahan dan bantuannya dalam pengerjaan Tugas Akhir ini.
4. Herninta Lagoon Fatika, selaku kerabat terdekat saya yang selalu menemani saya serta memberikan dukungan dan semangat yang tulus dalam suka dan duka mengerjakan Tugas Akhir ini.
5. Alcredo Simanjuntak dan Irham Muhammad Fadhil, selaku kerabat saya yang selalu membantu dan memberikan arahan saat mengerjakan Tugas Akhir
6. Angkatan 2016 Informatika ITS (C20) yang telah memberikan dukungan saat pengerjaan Tugas Akhir ini.
7. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa masih terdapat kekurangan, kesalahan, maupun kelalaian yang telah penulis lakukan. Oleh karena itu, saran dan kritik yang membangun sangat dibutuhkan untuk penyempurnaan Tugas Akhir ini.

Surabaya, Juni 2020

Fahrizal Naufal Ahmad

***(Halaman ini sengaja dikosongkan)***

# DAFTAR ISI

[**Abstrak vii**](#_Toc43758555)

[***Abstract* ix**](#_Toc43758556)

[**DAFTAR ISI xiv**](#_Toc43758557)

[**DAFTAR GAMBAR xvi**](#_Toc43758558)

[**DAFTAR TABEL xvii**](#_Toc43758559)

[**BAB I PENDAHULUAN 1**](#_Toc43758560)

[1.1 Latar Belakang 1](#_Toc43758561)

[1.2 Rumusan Masalah 2](#_Toc43758562)

[1.3 Batasan Permasalahan 2](#_Toc43758563)

[1.4 Tujuan 3](#_Toc43758564)

[1.5 Manfaat 3](#_Toc43758565)

[1.6 Metodologi 3](#_Toc43758566)

[1.6.1 Penyusunan Proposal Tugas Akhir 3](#_Toc43758567)

[1.6.2 Studi Literatur 4](#_Toc43758568)

[1.6.3 Analisis dan Desain Sistem 4](#_Toc43758569)

[1.6.4 Implementasi Sistem 4](#_Toc43758570)

[1.6.5 Pengujian dan Evaluasi 4](#_Toc43758571)

[1.6.6 Penyusunan Buku 5](#_Toc43758572)

[1.7 Sistematika Penulisan Laporan 5](#_Toc43758573)

[**BAB II TINJAUAN PUSTAKA 8**](#_Toc43758574)

[2.1 *Mobile Ad-hoc Networks* (MANETs) 8](#_Toc43758575)

[2.2 *Ad-hoc On-demand Distance Vector* (AODV) 10](#_Toc43758576)

[2.2.1 *Control Messages* 10](#_Toc43758577)

[2.2.2 *Route Discovery* 12](#_Toc43758578)

[2.2.3 *Route Maintenance* 14](#_Toc43758579)

[2.3 *K-Means Clustering* 15](#_Toc43758580)

[2.4 *Backup Routing* 16](#_Toc43758581)

[2.4.1 *Refined Route Request* (RRREQ) 17](#_Toc43758582)

[2.4.2 *Refined Route Reply* (RRREP) 18](#_Toc43758583)

[2.4.3 *Refined Route Error* (RRERR) 19](#_Toc43758584)

[**BAB III PERANCANGAN 25**](#_Toc43758585)

[3.1 Deskripsi Umum 25](#_Toc43758586)

[3.2 Perancangan Skenario Mobilitas 27](#_Toc43758587)

[3.3 Perancangan Modifikasi Protokol AODV 27](#_Toc43758588)

[3.3.1 Perancangan K-Means Clustering 28](#_Toc43758589)

[3.3.2 Perancangan Backup Routing 32](#_Toc43758590)

[3.4 Perancangan Simulasi pada NS-2 33](#_Toc43758591)

[3.5 Perancangan Metrik Analisis 34](#_Toc43758592)

[3.5.1 Packet Delivery Ratio (PDR) 34](#_Toc43758593)

[3.5.2 Average End-to-End Delay (E2E) 35](#_Toc43758594)

[3.5.3 Routing Overhead (RO) 35](#_Toc43758595)

[3.5.4 Average Hop Count (HC) 35](#_Toc43758596)

[**BAB IV IMPLEMENTASI 38**](#_Toc43758597)

[4.1 Implementasi Skenario *Grid* 38](#_Toc43758598)

[4.2 Implementasi Skenario *Real* 41](#_Toc43758599)

[4.3 Implementasi *K-Means Clustering* pada AODV 42](#_Toc43758600)

[4.3.1 Modifikasi Pengiriman RREQ 45](#_Toc43758601)

[4.3.2 Modifikasi Penerimaan RREQ 46](#_Toc43758602)

[4.3.3 Modifikasi Pengiriman RREP 47](#_Toc43758603)

[4.3.4 Modifikasi Penerimaan RREP 47](#_Toc43758604)

[4.3.5 Modifikasi Pengiriman Hello Messages 48](#_Toc43758605)

[4.4 Implementasi algoritma *Backup Routing* pada AODV 49](#_Toc43758606)

[4.4.1 Modifikasi Pengiriman RREQ 50](#_Toc43758607)

[4.4.2 Modifikasi Penerimaan RREQ 51](#_Toc43758608)

[4.5 Implementasi Simulasi pada NS-2 52](#_Toc43758609)

[4.6 Implementasi Metrik Analisis 53](#_Toc43758610)

[**BAB V UJI COBA DAN EVALUASI 55**](#_Toc43758611)

[5.1 Lingkungan Uji Coba 55](#_Toc43758612)

[5.2 Skenario Grid 55](#_Toc43758613)

[5.3 Skenario Real 61](#_Toc43758614)

[**BAB VI KESIMPULAN DAN SARAN 68**](#_Toc43758615)

[6.1 Kesimpulan 68](#_Toc43758616)

[6.2 Saran 69](#_Toc43758617)

[**DAFTAR PUSTAKA 71**](#_Toc43758618)

[**LAMPIRAN 72**](#_Toc43758619)

[A.1 Kode runCluster() 72](#_Toc43758620)

[A.2 Kode run() 72](#_Toc43758621)

[A.3 Kode getResult() 73](#_Toc43758622)

[A.4 Kode getClusterHead() 74](#_Toc43758623)

[A.5 Kode getClusterGateway() 75](#_Toc43758624)

[A.6 Kode sendRequest() 75](#_Toc43758625)

[A.7 Kode recvRequest() 79](#_Toc43758626)

[A.8 Kode sendReply() 84](#_Toc43758627)

[A.9 Kode recvReply() 86](#_Toc43758628)

[A.10 Kode sendHello() 89](#_Toc43758629)

[A.11 Kode konfigurasi skenario NS-2 90](#_Toc43758630)

[A.12 Kode konfigurasi awk 93](#_Toc43758631)

[**BIODATA PENULIS 96**](#_Toc43758597)

***(Halaman ini sengaja dikosongkan)***

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB I PENDAHULUAN

## **1.1 Latar Belakang**

Dalam kehidupan sehari-hari, banyak sekali aktivitas yang dilakukan, mulai dari pekerjaan, hiburan, olahraga, bersosialisasi, dan aktivitas-aktivitas lain yang sangatlah banyak. Bersosialisasi pun menjadi salah satu aktivitas yang sangatlah sering dilakukan. Maka dari itu muncul teknologi-teknologi yang berkembang dari aktivitas tersebut. Seperti telepon, ponsel pintar, handie talkie, dan teknologi-teknologi lain yang sudah berkembang di era informatika.

Salah satu teknologi yang cukup sering ditemui adalah Handie Talkie, sebuah perangkat nirkabel yang terhubung dengan perangkat lainnya tanpa menggunakan satu perangkat sebagai pusat menyalurkan data. Teknologi yang digunakan pada perangkat tersebut dinamakan Ad-hoc On demand Distance Vector (AODV).

AODV adalah distance vector routing protocol yang termasuk dalam klasifikasi reaktif routing protocol, yang hanya me-request sebuah rute saat dibutuhkan. AODV yang standar ini dikembangkankan oleh C. E. Perkins, E.M. Belding-Royer dan S. Das pada RFC 3561. Teknologi AODV pun pada akhirnya dikembangkan dan akhirnya muncul teknologi baru yang bisa digunakan pada perangkat bergerak, yaitu Mobile Ad-hoc Network (MANET).

Di dalam MANET, antara node yang berbeda dapat terhubung melalui transmisi wireless secara langsung, akan tetapi jika salah satu node diluar jangakauan transmisi maka membutuhkan node lain untuk meneruskan pesan. Oleh karena itu muncul skenario multi hop, dimana ada beberapa host yang berfungsi sebagai relay untuk meneruskan paket dari host sumber menuju kepada host target.

Sama seperti AODV, MANET juga menggunakan pesan route request (RREQ), route reply (RREP), and route error (RERR). Akan tetapi, karena sifat MANET yang sangat dinamis, maka routing tidak dapat dilakukan secara efektif. Beberapa penelitian juga melakukan evaluasi dari MANET. Dihasilkan bahwa jika MANET mengirim paket data melalui rute yang telah rusak, maka sistem tersebut tidak dapat memberikan rute cadangan langsung. Maka dari itu, perlu sekali dilakukan optimasi performa dari MANET.

Beberapa penelitian juga sudah memodifikasi AODV menjadi lebih efektif dengan melakukan menemukan rute-rute yang ada terlebih dahulu, baru kemudian dikirimkan paket data oleh sumber. Teknologi tersebut dinamakan AODV-ABR atau AODV-Advanced Backup Routing.

Berdasarkan permasalahan yang telah disebutkan, pada tugas akhir ini penulis akan melakukan modifikasi terhadap MANET dengan melakukan implementasi K-Means Clustering pada AODV Routing Protocol untuk meningkatkan rasio paket data terkirim dan mengurangi rasio Packet Loss Ratio, paket overhead, dan end-to-end delay.

## **1.2 Rumusan Masalah**

Dari permasalahan di atas, maka dapat dirumuskan beberapa masalah:

1. Bagaimana melakukan implementasi K-Means Clustering pada AODV Routing Protocol pada MANET?
2. Bagaimana peranan K-Means Clustering dalam AODV Based Backup Routing dalam mengurangi Packet Loss Ratio pada MANET?
3. Bagaimana peranan algoritma K-means dalam AODV Based Backup Routing memengaruhi performa MANET secara keseluruhan diukur dari Packet Delivery Ratio, End-to-end Delay, dan Routing Overhead

## **1.3 Batasan Permasalahan**

Permasalahan yang ada pada Tugas Akhir ini memiliki pembatasan sebagai berikut:

1. Jaringan yang digunakan adalah jaringan Mobile Adhoc Network (MANET).
2. Routing protocol yang diujicoba pada MANET adalah AODV.
3. Simulasi pengujian jaringan menggunakan Network Simulator 2 (NS-2).
4. Pembuatan skenario uji coba menggunakan Simulation of Urban Mobility (SUMO) dengan melihat indikator Packet Delivery Ratio, Packet loss ratio, End-to-end Delay, dan Routing Overhead sebagai penilaian.

## **1.4 Tujuan**

Tujuan pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Melakukan implementasi K-means dalam AODV routing protocol pada MANET.
2. Menganalisis peranan Algoritma K-means dalam AODV Based Backup Routing dalam mengurangi Packet Loss Ratio pada MANET.
3. Menganalisis performa AODV Based Backup Routing yang telah ditambah algoritma K-means dengan mengukur matriks Packet Delivery Ratio (PDR), End-to-end Delay, dan Routing Overhead.

## **1.5 Manfaat**

Tugas Akhir ini dapat dijadikan referensi ilmiah berupa hasil analisis performa AODV Routing Protocol pada MANET yang sudah dimodifikasi dengan metode AODV Based Backup Routing.

## **1.6 Metodologi**

Pembuatan Tugas Akhir ini dilakukan menggunakan beberapa Metode, yaitu:

### **1.6.1 Penyusunan Proposal Tugas Akhir**

Tahap awal untuk memulai pengerjaan tugas akhir adalah penyusunan proposal tugas akhir. Proposal ini akan menyajikan ringkasan dari metode dan teori dasar yang digunakan pada paper ini.

### **1.6.2 Studi Literatur**

Tahap Studi Literatur dimulai setelah proposal diterima oleh Dosen Pembimbing, pada tahap ini penulis akan mempelajari seluruh teori-teori yang akan dijadikan landasan pada pengerjaan Tugas Akhir dari buku dan jurnal-jurnal yang diterbitkan oleh badan jurnal internasional yang berhubangan dengan AODV, K-Means Clustering, dan Backup Routing.

### **1.6.3 Analisis dan Desain Sistem**

Pada tahap ini dilakukan perancangan alur kerja dari modifikasi AODV berupa flowchart, dan setelah dilakukan perancangan akan dilakukan implementasi dari AODV Based Backup Routing pada AODV Routing Protocol dalam MANET menggunakan Network Simulator 2.

### **1.6.4 Implementasi Sistem**

Implementasi merupakan tahap untuk membangun metodemetode yang telah diajukan pada proposal Tugas Akhir. Pada tahap ini dilakukan implementasi menggunakan NS-2 sebagai simulator, Bahasa C/C++ sebagai bahasa pemrograman, dan SUMO sebagai tool untuk uji coba dan mengimplentasikan desain sistem yang telah dirancang.

### **1.6.5 Pengujian dan Evaluasi**

Pengujian dilakukan dengan Network Simulator 2 dan akan menghasilkan trace file. Dari trace file tersebut akan dihitung packet delivery ratio, packet loss ratio, routing overhead, dan delivery delay untuk menguji performa routing protocol yang telah dimodifikasi.

### **1.6.6 Penyusunan Buku**

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

## **1.7 Sistematika Penulisan Laporan**

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan

Pada bagian ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

2. Bab II. Tinjauan Pustaka

Pada bagian ini berisi kajian teori atau penjelasan dari metode, algoritma, library, dan tools yang digunakan dalam penyusunan Tugas Akhir ini.

3. Bab III. Perancangan

Pada bagian ini berisi pembahasan mengenai perancangan beberapa skenario antara lain modifikasi, pengujian, dan perhitungan metrik analisis.

4. Bab IV. Implementasi

Pada bagian ini menjelaskan implementasi yang berbentuk kode program dari proses modifikasi, proses pengujian. Dan perhitungan metrik analisis.

5. Bab V. Uji Coba dan Evaluasi

Pada bagian ini berisi hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.

6. Bab VI. Kesimpulan dan Saran

Pada bagian ini berisi kesimpulan dari hasil uji coba yang dilakukan, masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan dari solusi yang ada.

7. Daftar Pustaka

Pada bagian ini berisi daftar pustaka yang dijadikan literatur dalam pengerjaan Tugas Akhir.

8. Lampiran

Pada bagian ini terdapat tabel-tabel yang berisi data hasil uji coba dan beberapa kode program.

***(Halaman ini sengaja dikosongkan)***

# BAB II TINJAUAN PUSTAKA

Bab ini berisi penjelasan mengenai teori-teori dasar yang berkaitan dengan pengimplementasian perangkat lunak dan penunjangnya. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap *routing protocol*, tools, serta definisi yang digunakan dalam pembuatan Tugas Akhir.

## **2.1 *Mobile Ad-hoc Networks* (MANETs)**

Mobile Ad hoc Network(MANET) merupakan sebuah jaringan yang terbentuk dari beberapa *node* yang bergerak bebas dan dinamis. MANET memungkinkan terjadinya komunikasi jaringan tanpa bergantung pada ketersedian infrastruktur jaringan yang tetap [1]. Dalam MANET, setiap *node* yang ada dapat bertindak sebagai host maupun router. Setiap *node* juga dapat saling berkomunikasi antara satu sama lain walaupun tidak ada *access point*. Perangkat pada MANET harus dapat mendeteksi lokasi atau keberadaan dari setiap perangkat yang ada. MANET dapat dimodifikasi sesuai dengan kebutuhan di lapangan, karena hampir tidak membutuhkan support dari infrastruktur yang telah ada. Antar node akan terhubung sebagai jaringan ad hoc sebagai autonomous system of mobile hosts (MH) yang juga bertindak sebagai router dan terhubung menggunakan jaringan wireless. Hal ini kontras dengan jaringan seluler single hop yang membutuhkan BTS ( Base Tranceiver Station) sebagai akses poin. Dalam jaringan seluler, komunikasi antar mobile node tergantung terhadap wired backbone dan dukungan sinyal dari BTS. Di dalam MANET tidak dibutuhkan infrastruktur dan topologi jaringan dapat berubah secara dinamis mengikuti perubahan node karena node dapat bergerak secara bebas. MANET mempunyai beberapa karakteristik, yaitu:

1. Topologi yang Dinamis:

*Node* pada MANET dapat bergerak secara bebas dan mampu bergerak ke segala arah layaknya sebuah perangkat *mobile*. Hal ini menyebabkan topologi berubah secara acak dan tidak mempunyai pola.

1. Keterbatasan Energi:

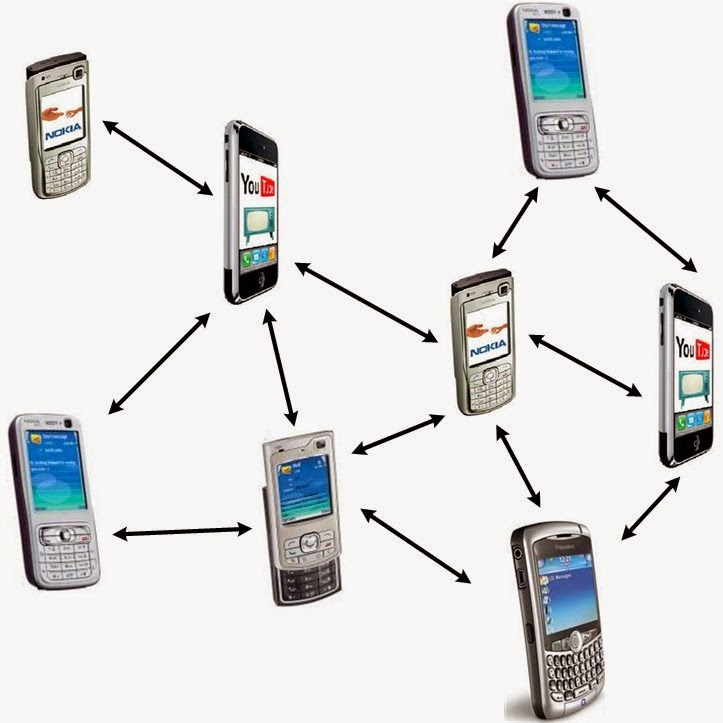
Karena bersifat *mobile,* maka setiap perangkat yang ada tidak tercolok langsung ke sambungan listrik. Melainkan menggunakan baterai yang tentu mempunyai batas energi dan umur baterai.

1. Keterbatasan *bandwith*:

*Link* pada *wireless* jelas memiliki kapasitas yang lebih sedikit daripada perangkat berkabel. Selain itu seringkali terjadi kemacetan ketika menggunakan perangkat *wireless*.

1. Keterbatasan Keamanan:

Jaringan nirkabel seringkali menjadi sasaran empuk bagi penyerang keamanan. Contoh seperti *spoofing, denial of services, eavesdropping,* harus menjadi pertimbangan utama saat membuat suatu jaringan.



**Gambar 2.1** Ilustrasi Jaringan MANETs

## **2.2 *Ad-hoc On-demand Distance Vector* (AODV)**

AODV merupakan perpaduan antara DSR dan DSDV. AODV mengambil karakteristik DSR yaitu melakukan route discovery bila dibutuhkan. Perbedaan AODV dan DSR adalah AODV menggunakan routing table tradisional yaitu satu entri per tujuan [2]. Seperti DSDV, AODV menjamin tidak akan terjadi loops. AODV menggunakan komunikasi broadcast untuk route discovery dan komunikasi unicast untuk route reply. Sedangkan pada DSR digunakan routing yang telah ditentukan oleh node asal, bukan rute yang ditentukan oleh node perantara yang ada di antara node asal dan node tujuan. Node perantara memiliki route cache yang akan menyimpan informasi untuk keperluan routing. AODV sendiri dapat dikatakan sebuah *routing protocol* yang sangat direkomendasikan untuk sebuah MANET. Karena AODV dapat mengatasi keterbatasan *bandwith* yang dimiliki oleh MANET dengan sangat baik. Salah satu keunggulan dari AODV adalah *routing protocol* tersebut tidak menyimpan informasi tentang *route* yang sudah tidak aktif dan penggunaan *sequence* *number* yang disisipkan pada setiap paket oleh setiap *node* yang mengirimkan paket tersebut yang bertujuan untuk menggantikan cached *route*s sekaligus memastikan bahwa *route*s yang ada pada *routing table* adalah *route*s yang terbaik berdasarkan *sequence* *number* dan *timestamps*.

AODV sendiri mempunyai 2 fase utama, yaitu *Route Discovery* dan *Route Maintenance.* Pada setiap fase, proses ini melibatkan beberapa jenis paket yang disebut *control messages*.

### **2.2.1 *Control Messages***

*Control Messages* adalah paket untuk mengatur proses *route discovery* dan *route maintenance. Control Messages* sendiri mempunyai 4 jenis, yaitu *Route Request, Route Reply, Route Error, Hello Messages.*

#### **2.2.1.1 Route Request (RREQ)**

RREQ adalah jenis paket ketika *source node* dihubungkan ke *destination node*. *Source node* sendiri akan menyebarkan *broadcast* untuk RREQ terhadap *node* di sekitarnya. RREQ sendiri memiliki beberapa *fields* seperti yang tertera pada Tabel 2.1.

|  |
| --- |
| *Source Address* |
| *Request ID* |
| *Source Sequences Number* |
| *Destination Address* |
| *Destination Sequences Number* |
| *Hop Count* |

**Tabel 2.1** Format dari RREQ

Setiap kali *source node* mengirimkan RREQ baru, maka *request id* pasti akan bertambah, sehingga dapat dikatakan bahwa RREQ yang dikirimkan adalah unik sesuai dengan *request id* dan *source address*. *Node* yang menerima RREQ dengan identitas parameter yang sama akan membuang RREQ tersebut, sedangkan yang diterima, *hop count* dan *sequence* *number* RREQ tersebut akan dibandingkan dengan informasi yang ada di dalam *node* tersebut.

#### **2.2.1.2 Route Reply (RREP)**

RREP adalah jenis paket yang digunakan ketika node tersebut adalah node tujuan atau menyimpan sebuah rute menuju node tujuan. Setelah itu node tersebut akan melakukan *unicast* RREP ke pada node sumber atau *source node.* *Fields* pada RREP ini tercantum pada Tabel 2.2.

|  |
| --- |
| *Source* Address |
| *Destination* Address |
| *Destination* *Sequence* No |
| *Hop Count* |
| Life Time |

**Tabel 2.2** Format dari RREP

*Unicast* dapat dilakukan karena setiap node pada suatu rute memiliki data *route* untuk kembali ke *source node.* Proses ini dinamakan *Reverse Path Setup* (RPS). Sedangkan dalam pengiriman RREP secara *unicast* setiap *node* yang berada pada *route* tersebut menyimpan informasi darimana *node* tersebut mendapatkan paket RREP atau yang disebut juga dengan proses *Forward Path Setup*(FPS).

#### **2.2.1.3 Route Error (RERR)**

RERR adalah jenis paket yang digunakan untuk melakukan pengawasan terhadap beberapa *node* yang bertindak sebagai tetangga dari suatu *node*. Ketika suatu *route* tidak bisa digunakan karena adanya masalah pada suatu *node* di dalam *route* tersebut, maka RERR akan dikirimkan untuk memberikan informasi bahwa *route* tersebut telah *invalid*. Tujuan pengiriman RERR adalah untuk mencegah pengiriman paket kembali melalui *route* tersebut dan sehingga perlu dilakukan proses *route* *discovery* kembali.

#### **2.2.1.4 Hello Messages**

Setiap node dapat mengetahui tetangganya dengan mengirimkan local broadcast atau juga yang disebut dengan Hello Messages. Node tetangga adalah node yang diasumsikan dapat berkomunikasi secara langsung dengan suatu node tanpa melalui suatu intermediate node.

### **2.2.2 *Route Discovery***

*Route* *discovery* adalah proses pencarian suatu *route* menuju *destination* oleh *source* *node*. Dalam prosesnya *node* akan menggunakan RREQ *message* dan RREP *message* untuk mencari *route* yang diinginkan oleh *source* *node*. Ketika suatu *node* mengirimkan RREQ, *node* penerima mungkin saja mendapatkan RREQ yang sama kembali dari *node* tetangganya, sehingga untuk mencegah *infinite cycles* setiap *node* memiliki *buffer*, yang berisi daftar RREQ yang telah dilakukan *broadcast*. Sebelum RREQ dikirim, *buffer* akan selalu diperiksa untuk memastikan RREQ yang sama tidak akan dikirimkan lagi. Setiap intermediate *node* akan melakukan penambahan *sequence* *number* untuk memastikan bahwa *route* akan selalu diperbaharui dengan informasi terbaru. Pada setiap RREQ *message* akan dimasukan *source* *node* *sequence* *number* dan *destination* *node* *sequence* *number*. Gambar 2.1 akan menjelaskan bagaimana suatu *node* melakukan proses terhadap suatu paket *message* yang diterimanya.

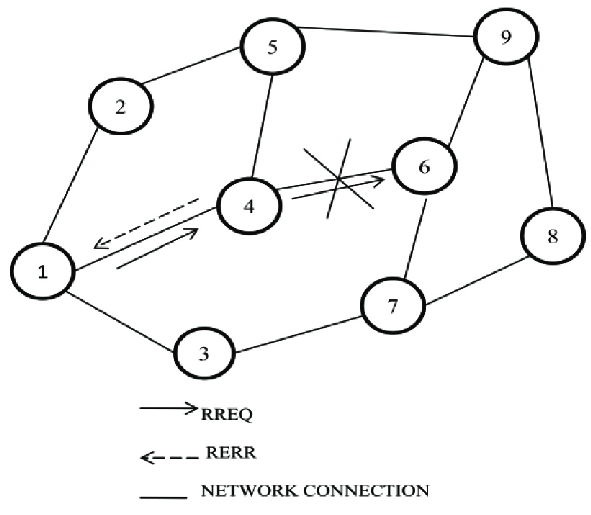
![A close up of text on a white surface

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDiRXhpZgAATU0AKgAAAAgABAE7AAIAAAAIAAAISodpAAQAAAABAAAIUpydAAEAAAAQAAAQyuocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEFsY3JlZG8AAAWQAwACAAAAFAAAEKCQBAACAAAAFAAAELSSkQACAAAAAzI3AACSkgACAAAAAzI3AADqHAAHAAAIDAAACJQAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDE5OjExOjIzIDAzOjUxOjAxADIwMTk6MTE6MjMgMDM6NTE6MDEAAABBAGwAYwByAGUAZABvAAAA/+ELGmh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTEtMjNUMDM6NTE6MDEuMjY2PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkFsY3JlZG88L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgCoQHYAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAoopCwFAC5pM81HPPFBC0k0ixxqMs7nAUe9c1eeOdNjuJINNgvNYmjP7yOxiDbBxzuYqpHIHBNAHTeZz2xnGc04thgK48zeNNTbbHZ6fpMTfPHcPMZ5FHZWjwBnHXnj3pR4Z8RX7bdb8TyhF5T+zYvszZ/2jk5FAHW7yT8o+uakBrjz4Cjn/AHd/rur31s3EltcXRaOUejClX4YeDcH/AIkdv+RoA6/NGa5D/hWPg7/oB24/Okb4Z+DQM/2Jbce2aAOuLGkWTLEH8PeuRXwDaQLs0jVtT0q1H3bayuSkanuQKD4c8Q2B8vR/FEpjbl/7Ri+0Nn2ORge1AHWGR9xwuFHXNSZ/OuPNz400wbZ7Gx1aCM75LiKYwyOO4WLaRn0+YZ9qtWfjfTJbhbfUYrvR7iQ5SK/iC7l/vblJUDqOSDxQB0+aKghuYp41khdZEYAhlOQc+hqbNAC0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFZGua3b6Dp/2q7DuGlWKNIxlndjhVH1Na9crflT8QtMS95gNtI1sr/dM4IJI/wBoJk/SgCOLw9f65Kt14qnzEpzHp1uSIl/3z1Y4OCDxVTxZqzeD7PTrbRItNtFnd4910CsaAIWwNpHJIrtweKyNW0WLVdU064uFjkjs3dzHIgYMWUqOvpmgDLt/HGm/Zgb0SRTx2y3FwFQlY9wyBn1PYVKvjnSGsoriP7S/mSGIRCE+YCOuV7Dmm6j4TF5/a3lzrG1/5RjJTiMoCB9RVGLwjq1vNHqMWqQ/2r5rNNMYPkZCBwF7dKANU+MtJXUp7Iyyebbxs7kodp2jJAPr7VCvjnSngtJgt3i6kMcQ8g7s8ckdhz1rHg+HT23iW91SGa1Y3BkdTLGzMruMHvjHtimJ4A1Mada239o2+YbnziRC3ycj7nOR+PFAG5qfiqLR9cmgvgFtIrJblnAJbLSFAAPrio38Z2DWUV/b73t2jmdomiIl/d9QB2p2reFf7V1Brlrkput4YCGXOdkvmZ/HGKWbwp5l68yT7Q32jjb080AfpjNADYPF9rLbreeXKlqbRbkxeUxl5Yjp6U2Hx3plxd6dDBHcut8XVW8ojYyEAhh261VufBt/eaZJDPfp9oNklp5gQhSFfdkgc89Kh0jwLdaPa6fHbX0StbTTvLsjO11lILKMnI6cGgDVi8caLPBdyJLNttMFyYj8+TgbP73NLo17aeMNHuHurNGt/PeERypyQp7g9DXOaR8MX0e1uFt7u3SX92beYRsWUo275snkV13hzR7nSdPkS9uVubied55JETaNzHJwOwoAypvDlzoErXvhfLbjmeyuJGaNkHOEH8Ldh255rodL1GHVLGO7ts7X4Kn7ykHBU+4Iq9+Nc1pYUeOdX+xf8evkxecI/uC4yd3/AALZsz/jQB09FFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABWPrmi2+s2ypOzxywvvhmiba8beoP8/UVsUgUDPvQByUXiG+0KVbXxXb7I2O2PUYMtC3+/3Q4GSSNvPBroLS9tr63WeynjuIXyUeNgwIqzcW8VxA0U6LJG4wyuoII9wa5u78C6dJcyXGl3d9os8h/ePp8oXePTa6soH0AoA6RcHnrT+K48WnjHRc/Yryz1y0j+WOC5Tybhh6vMDtyPZOfalHinxBYMH1zwpNsbiP+y5/tbZ/2htXaPfmgDsMCjHFcifHm35rjw3r0EQ5eWSxwqDuxO7gDrR/wtLwUP8AmYbT/vo/4UAdbgCjAznHNcifip4IEip/wkljuYZC7+SKVvih4LIx/wAJDZ/99H/CgWp11GBXI/8ACwIpfn0/QtbvrduY7m2s90cq9mU55B9aD4o8Q33z6N4UmMS8P/aU/wBlfPsu1sj3z+FAzquMgY44qve39tp9u897cRW8UYyzyOFA/OuaNj4w1o/6ffWmi2snD29ohluEx3WckL1/2P8AGrVn4G02O6W51S4vdauoz+7m1CUHav8Ad2oqoQOTypPPXpQBBL4hv9fka28KQZgzsl1G4BWJfUIMZZscg429s1s6Fo9voOlrYWhkZA7yM0jbizsxdiT7kmtGGCKCBYoY1jjQYVEAAA9gKkC4NAC0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRiiigBhTPU0bOfr1p9FADNp7GlwtOph/KgDzXVgP8AhpLQTj/mBT54/wCmhr0wAf5FfGOpeE9ck+OreGDeXBmkvCqz7yp8hjvJB7Dbn8cV9kWsSW1pDboxZYkCAnqQBjNAEpVvWkEfzH5jg9s08e1FADdvv3zTsUUUAFFFFABRRmigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKTvSnpXJ674ruNJ8QR2UdgJ4AsTzzCZQyB3KDCdW59KAOrorEi8U6JNdXNtFqdu01qC06BwTEAcHP4mk1HxVo+kXEEN/fRxPNMIVBP8RUkfoKlgblFZVtr+mXmpS6fa30M13BzNCrfMg+lYul+MJLvVbiC9tVtbZPNMc/nq24RthiQOV6d6oDr6Kwf+Ew0D+zP7QOr2otDJ5Qm8z5d3Xb9cVcXWNPkQvHdxuoVCWD9m+7+dAGlS1hnxToomvITqluJLFS9yPM/1QHXdUVv4y0K61G0srbUYpZbxDJb7GyHAwDj86AOhopqn5RmnUAFFFFABRRTD3OaAH1zvi/xdpPgzRpNS1u4WGPBVF7yNjgCsXxp8SrDw1MdKsN2pa/MhW30+2XfIXxkbgOg+tYvh74cXms69F4s+IM/2zUSPMi0770FsCuNu08ZFAHJWeh+N/GfiRfiNZ2kGj3UCL9itHTJu4schz15HA9K9F8E/Eey8TbbHUVGm69GMXGnS5VlOT93PUEDI9q7xFVY1CgAAYAAwAK4Tx98NrPxdsv7NhYa7bMpttRjBDJg9DjqO1AHdj2pwryXwl8Sb7SNRXwv8SEOnahFlYNQnO2K+AbaCp6EnI+terIc4PX8aAJKKKKACkpaRulABxRXj2nXt1o3iTW9SEMiP5uoNE73BdLnFxgDaThNnAxXU6f4m1OG8is9UntZm+1LFJcQDEe1o2YcnuCv60AdxRXma+MNYjmvdTlngezh0dLxbVVJd3zJ938VFPg8Q6rrfw38QzalDJFNFDKqPsMG8eXnjPI+tT1A9JorzfRLvUdMhsrHTbeCzRraW7uI5Lg3BO0qAFbJ65B9qVvFPiMWVkzT6fFJJZG/dnBwyZXEa/7XzdfaqA9Hpa5jw7q+o6vf3ss/lR2UTCOOLB8wNtVuT/wI101AC0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAIa5HxB4RGr64dUi8pbuCOJbaVicoVcls+xBNdeehrnJ/E8cHjJtClt2UCyN0LkuNpwSNmPoCfwoA5oeCNXn+1WsjWMdsq3P2adHYySNK+7EgxgYyehPSriaD4imVbu6j0z7at8k4hWV2jEaxlDhtmQefTFXNI+IGkXmhw6nqdxDpUdxcSQRLczAeZtcqCD3zgHHbNas3irQodQuLKXVbNbm2iM88RmUNGnHzEdhyOTQBzfhvwXqOkeLrjUbuRJrZmlMMgumLDe2ceXswPwY0ybwBMsF+dPlhtp9RhuI7mRWI3l2JQ9OozjNbr+N/DCw2kz69YCK8yts5uFxMQcHaT1wSO9Tv4s0GPVpdLk1ezF9ChkktjModFC7skE8cc0AcRa+Cde0vS/tESWMt3HPJI0c15JKjq0WzO7ywQR2GO3WrWleHNdfTrA2y2kdrcW1t9pM7sskZiOflUKQfxIrqv+En0u58N3WsaVdwahbW4Yl4G3qWXtkd+lZv8AwmF5DrzWVzpQFupERu0uBh5fJEuPLxkemc0AYVr4B1eyvtUmYW14kyTCHzb5xne+7lRH8v1BNbGjaBrOn3GkTXbWUxto5YrgmRmKIxUja20bj8vUgVBB4/vptNMp8P7bxzCbe1+2qRIsv3SW24X6EGrN9401CzBzoEsv2WJZdQ2XKn7OG6AHGJOAemOnvQB2S7ttPFY2jao2oXF9FIVH2eUbCB1RlyD+hrZoBC0GkLYqnqOqWmlWMt7qNwlrbQrukllbCqM9Se3/ANegCw0m3OcfXPSvKvGPxLvdR1STwr8N4BqesEOlzPu2x2eMDOTwx5zgVRufEviP4tXt5o3g9ZNI8OqzpNrpDFrnawBWI9s56/qK61rHQfhX4fhfSNJyk06QzShsyuXPLszcnnnHT2FADPAXw1svCTyahqFydW1q4YvLqE6/NkgAqvtxXd7Qa5lvHOiWvmtq1/bafGty1vG9xMoErAAnbn61aj8Y6FLqk+mpqdv9rt1V3QuBhW6H9RQBvUmB3FZl34g0uxljju9Qt4Xkl8lFkkALP/dGep9qLHxDpOp3VzbadqNvcz2p2zpDIGaI/wC0B0oAp+KfBuieMNP+ya5ZiZVB2SDh4+mSp7HivN4vE+v/AAiu00/xcs+s6BcSO1tqsOWkgXcABKD2AbOQfpmu51v4i+H9L0m8vINRs7ySzkWJ4VuVXa5OArE/d6H8q25bS117Q2t9Ttklt7yH99ATuVgw5GR169RQBYs9Rt9Qt0ubG4juIX6PG2QauCvFb3w94g+EN/dav4PLal4TUST3Oiu43Ww2glkY5YgYJx6ZznrXpvhfxZpPi7R49T0S7W4hb5XAb5omwDtYdiARx70Ab1IaQSAge9B6HFAGcdK050KtaRFN7sQR/E5Jc/iSc1VHhTQv7L/s8aXbi0L+Z5O043eteWz6pd2+j+IdLTUbj7Tf311NA/mEtFGksm/YeoA2qPbcK6mz8Y6vJ4sj0q1sJZbGMrDK32OQksUB3edu2D6YoA6//hHdI3xkafADHCYE+XpGf4fpzRa+HNHsNMnsLXT4YbS4BEsSj5XyMHP4Vxv/AAlnidfD0k0gtBfQ3RS4UafMRBHtHPl79zDOfmBAxUzeNNTTxklihgudNkRsyx2UihGERkx52/aeg4x3oA6yw8OaTpcbJp2nw26MrKQg7EgkfjgVHe+H9GvY7ZNQsoJEtSBArjhPYflXAL8QPFcelrLPYAz3Vz5Vps0ubOBGXY+Xvyw4wCCB36Cui1G+utQsPDk14j2ktw5kngZTGdwt5Dgg5I+YA4zQB1sMFvbmQwqieY25sfxHpn9Km35AwRnGRg9a8Q0xLweEWiNndCW4FiZLJ9UkJud3WQTFsoG7getal1bS2vjKzH2O6sobeOyBm/tOSVbVmd8qy5xJnAG45xigD1pp1RS8jhVAySTwKlB4ryfRrd2vdX+129xAJ7e6bc948q3eHIyqscRlcdB616ZpMrT6PZyyDDyQozD0JAoAu0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABSFsGlqKQblYHkEUAPDArwc1xfijwjf6xqkl7YXEcEhWFUcnkKCwkB47o7Ae+K5b4W+FNO8S/D+y1TV3vLi6mlmDv9qcZ2zOo4B9AK7H/AIVt4d/553f/AIGSf40Ac1qnw71t1i/s662hZ7nfGl5JbgxySl1yUHOB2rTk8M69/al48cVr9hktTEsDXchEr4ADMNuAePvDmtH/AIVr4e4/d3fH/T5J/jR/wrTw7/zzu/8AwMk/xoA5vV/Bfi7UvDdnp5v13hZkuAuoSrw2NuWAzJgA/erNfT7vU/F2uaXCZlkuYJo0ZmmVbcmLAYDbsGSoGVYnnpXb/wDCtfDuf9Xef+Bkn+NM/wCFZ+HQ24RXYPr9rk/xoAztK8M6rpXgvV4L4mS9uSXU/a5LgnCAD5nGR0xVgeDjLrsniG0jt47u7Yb3IIaSEwKmxuOuRnFW/wDhWvh7aB5d3/4GSf40f8K08OdPLu//AAMk/wAaAOdf4ZzWfhoaXptnpk1sVtnms7gFYZJY+HJwOjcnp1qa78L+KbfS7q20S00m3S+t0ilgErqtuVBH7s7eQQRwQBxW5/wrXw7/AM87v/wMk/xpf+Fa+Hv+ed3/AOBkn+NAD/CUMhuNQu5bSS1MhigEM2Mr5an0JH8X6V1R6VyH/CtfDv8Azzu//AyT/GlPw18O45jvP/AyT/GgDT8TeKdK8KaS9/rV2sEQbYvBZnc9FCjkkmvLLXQvEvxg1SO/8Ux3Wi+FYipg0tmw15husi/h3/ka2PFnwbgvpLPUPDl/NaX2nyiaGK4czQyMDn5ge+Bjg96v+DviZHqVxHoPiu3GjeI0JDWzDEcnzEAxk9c+lAHc2GmWul6fDZadbR2ltCu2OGFAqoPQAdKqa/pTavBaxEpsjuEkkDfxKAcj8c1sD7ox6Vz/AIr8Sjwxp0F1JavdCW4SHCMARu/i+goA4mfwD4mh0GXTrLUEKSXkzkreSQkxsAFBZRkkc8dK37rw1rLR3kFs1m0MsEBR5Xbd50eODx9046/pWmfF1mPEl1pToUjtbUXEl0zfKAT0x/nrVn/hLND+zW851O3MVzIUicH77Z6D86AMbTPD2rnUrfUNXFok4mneRbdywQyKAMEgelVfCPgvU9H1C9OqTtPE8TwwyG9klIVjn7rDC9e1dJL4l0dJLpH1GFDZj9+Af9WD602TxboMOnw3k2qW6W85Kxy7vlYjggUAc1F4P1me3EF4LCGO1jSG3MLkmRRKGLN8owcDtXoAX5RWda67pl3qT2FtexS3MabzEp5C+taY6daAIyhwRgYzXmPi/wCH2rafql14n+GlyLLWZSXvLUtiG9AU7QV6bs569zmvUj0Ncn448eaR4F0s3eqO0s0m5YLWIZeVsZxjsOnPbNAGd4A+Jdh4uR7C8jk0zXbYKtxYXQ2OTtyWQd16+/GcV3g+7XiXhv4d6n448VDxx46QWEjRqtpY2jFf3flkZdhzn5jXoA+Gvh3H+rvP/AyT/GgDbbQNNMhb7Bb52yLu8sZxIQXH/AiAT6kVXm8I6Dcaouoz6PaPeIuxZ2iUuFxjGetZv/CtPDv/ADzvP/AyT/Gj/hWnh3/nnef+Bkn+NAFgfD/wqNPFj/wj+n/ZRL5wh+zrtD9N2MdeBU8/g/RXvpdRi061i1KSNkF6sQ81crtznr04rP8A+FaeHP8Annef+Bkn+NH/AArTw5/zzu//AAMk/wAaAKeh/DLTLHT7uz1e3stRhuJBIYjZpHHkdG2Dgt7nmuhuPD1q9vp0FvHHbxadKHgjRBtUbWTaB2GGNZf/AArXw9/zzu//AAMk/wAaT/hWnh4/8s7v/wADJP8AGgDUu/C2i6hara3uk2k1ssSwrE8SlVRcYXGOgwOPan23hrSLO1+y22mW0cGxU8pYwF2qcqMenJ/Osn/hWvh4/wDLO7/8DJP8aP8AhWvh7/nnd/8AgZJ/jQBsX2h291ZPBEiQsVdUkVRlN/3sfXNaESrFEqLwEG0D2HFcv/wrXw92ju//AAMk/wAa4vxvoFp4V8T+CX0aW6g+165FBOGuHYOhBOCCfagD2IGikHaloAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACmv8AdP0p1Nf7poA8++Bv/JItM/67XP8A6Pkr0OvPfgb/AMki0z/rtc/+j5K9CoAKKKKACk70tGKAEpaMUUAFFFFABiiiigAwK4zx18PtM8aWZM4+z6lCAbXUIhiWAgg8H6jpXZ0m0elAHkHh7x5rPgvU18PfEgOLcbhaa04O24G75dx6A4P6V3viPSD4htLOOF08tLhJWOeCuD0/OtDXdB07xHo82l6vbLcWkww0bfoR6V5PJd+I/g1M4uIbjXvCcszyJJGS8tgvGA2f4cfyxQBvf8IDqr6PcRzXitezWzI8qHaS2/cAD6YAqODwVrVtpNlDFBEt1FdGV5ZLsuUGR0yOQcdK7vRtd07X7Bb7RruO8t36PEc4PofQ81qbVPUA0Aec3HhjxC8usNb/AGeCO7QCLEv32yMnp8mQDXMa3o11oGlaPZ6mjSyfaJJWTzmYNufO0sBz16Gvbti+lNKKeooA878M+D9RsfGD6tcM5tZN8kf+kH5d5B27OnFehZwOtMd/LyW2oq5yScACvIvEvj/VfG+p3HhX4YJ9oZVePUNTbKJbjcFyjfxH73T8KAN3xp8TY9Lv5PDvhiI6p4klR1itohuETYGC355/DFReFfhns1keKPGVy2qa/LlyG/1NvuUAqqdOMEZrc8D/AA90vwTav9nzd30zs819ON0rlsZ+brjgV1wVfSgAUAKMDA7U+kooAWikooAWikpaACiiigAooooAK8x+LX/IxfD7/sY4v/QTXp1eY/Fr/kYvh9/2McX/AKCaAPTqKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigApr/dNOpr/dP0oA8/8Agb/ySLTP+u1z/wCj5K9Crz34G/8AJItM/wCu1z/6Pkr0KgAooooAKKKKACiiigAooooAKKKKACiiigAIzUFzaQXVtJBcxrLFKpR0YZDA9RU9BGRzQB43qfgjWvh1rFz4i+HTifTQGmuvD8h2qcDkxtzgnngj8673wf450vxnpYuNMkKTxgC5tZBiS3cj7rCujKHtXnXjf4b3N1eyeJPAt62jeJAd0kqN+7vAo4SRD8pOe5H1oA9GEmcZGKztb17T/D2kzalrFzHaWkON8shwBkgfzNeXaT8eLKytDpnjWwvbDxHABH9kFu5N0+OCmBxuPApdL8E658S9TtfEfxCkls9OUh7bw8gKIME4MwPUnrg8846cUAV1vvEfxmaa2tVk8P8AhDcSLzG6e/Cv0HI2Ajr17V3Ojaj4O8M6inhbSHt7K73AfZ1U5ZiC3LEdTXUW9lDaW0dvaQpBBEoWOKJQqIB0AA6CvJNb0PXD8QbzU7eK5ns/7ShYW6wfKW8ohZgQM/KeDzjnmgD2AEDqef5UhuUWRUdlV2ztUnk49K8R0uPxJHYNdRReIMwxWsmoxzecWlmWbL+UGPI2Donyn0q7HY6jqev6Xrd1Zayp+33QhJE6mMMFMW9AflXOc5AHXNAHsLzrFG0kjBUUEsSegFZOh+MNF8SSzR6NfJctCMuFBBxnqM9a8++Htr4g+0au2rSXvmNaMskE8NwQ0mW+ZWlJXPbCYX2rnrbQvFOl+HZ/LivnuptPtzFJ5DRtDEJv3kA8vB3Y54+bHc0Ae9b+ef8AIqva6nbXdxcwW8okktXCTAfwNjOK8Hu31kWmn2F7da5cGSzvpbeC0S6ikRxIgi3AnzCq7sZckEdc1ra+viTQtK1LW4Y7sagt1HGYwSq3JkgCE+hw+ORQB7YG4p1UdLtDZaXbW7yvL5UaqXkYlmwOpJq7u9KAHUUUUAFFFFABXmPxa/5GL4ff9jHF/wCgmvTq8x+Lf/IxfD7/ALGOL/0E0AenUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRSZoAWigUUAFFFFABTHP3vYU+vO/jTrmv+G/AUmreGpVimt5kMzMobEZODwfcigBfgcf+LRaZ/12ucf9/wB69Dr5x/Zq8R+I9TvLnS7ifdothCzBNgBWR3LdevdjX0dQAUUUUAFFFFABRRRQAUUUUAFFFFABRSZpaACiiigAphG7OaeaYSR0/WgDi/EGiadd/Evwzf3NlFLcotyFldckbUBH6k12W0A89a+a/HXxs8S+HPiN9lutG09pdIkkWI5kAlWRQAx59PSvoHw3fX2p+GtPvdVt47a8uIFklijJKoSM4GeelAGtTCMAgnH40815Rr2j6rqPjCZzNqEdudUhhCwyMq+QYiWPH+1jmgD1PGew570YDE5HT9a8IiHi/wAzQI7ma6hCQQCOaRJGIcSHduCkDkYzuzV27l1tZLvy4r64C6oskZxKv2gHqnB+ULxyODQB7VgDjgDtQR6Hivn3U9S8QRSawZIbrT7e4MKBS8pw4n5GSc5K/wB3FWrf+2pbjRRpWo3M1kY02XLLK+JRMfMBAPr8vzZwKAPUtW1/QNK19p9VHlXFlahmumjLeWkjABeOckqO3aqupeNfBc1pZzapqMMkMrtNb74XOGjYbjjbldpYdcVy3jSwu7/XNdt7cOZ5V05o2RNzIA5ywHoDk81y2reFdcn2wQ28/wBotrfVBcyiIbbstJEcdPl3gEjHpQB7FN468O29zc28uojzbRS82I3IQBdxJYLjpzSWnjjRNQuLNNNuTdC6uGt1dUICuE34YNgjj2ry+aC8a38Y28MeowfbrWVrWz+ygo5+zAAlyM5yMYrVj8M6hpV5o13fywjUdS1mCVkt1KoiJEARg852qc0Az2AU4UlLQAUUUUABrzH4tc+Ivh9/2McX/oJr01uRxXyh8c/F/ivSPidHay3Kra2M8eoaYoRTs+XAJ9fmDcUAfV+aWuW+H11rN94D0q78Rur6jcQebMyqADuYleB/skV1FAC0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFc94w8XWXgzRBqmpRTywmZYQsCF2LNnoB9K6Gub8XDMmgg99YhB/75egDix+0J4bHXS9Z/8AAJ/8KP8AhoXw3/0C9a/8An/wr1MogH3F/Ks5tZ09VQkqA9wbYHZn94M5H6GgDz3/AIaD8N/9AvWv/AJ/8KP+Gg/DY/5hetf+AT/4V6gHhO7/AFfy9enFVLLU7K/WVrZ42WKVomyuPmU4IHrzQB51/wANCeGv+gXrX/gE/wDhWdrXxr8Ka9ot5pl7pWtNbXkLwSL9ibgMuCRxweeK9fSaB5HjUxM8Zw4GMrx3qnq2qWOj6bLfXufIQqP3URdiScDAGSckgUAeAfCXx/4e8AeETY3Gmau17PcPNOwtGbPZQDjptA49c13o/aD8Njrpetf+AT/4V6DpmqWGsael7Z/6mQkASRlGBBwQVbkHjpV5niVSf3eB16f56c0AeYf8NCeGv+gXrX/gE/8AhR/w0H4bP/ML1r/wCf8Awr02Oe2liSSNomRxlWGMGnF4goZgqhhkZXFAHmH/AA0L4b/6Betf+AT/AOFH/DQnhv8A6Betf+AL/wCFen+ZCFyTH9eKUtGCARHz0HGTQBzPgv4hab46F2dMguoDaFQ6XMRjY7s4IB+hrrh0rmrJf+LiasAAANOtTgdvnmrpR0oAKKKKACg8Cig8igDzfX/jRoXh3XrzSbqy1Oee0cJI0FszrkqD1H1rP/4aD8Nn/mF61/4BP/hXY+FUVr7xGSoJGrvjI/6ZRVZv/EFtYLqLSWzEaesbPgD5t/TFAHC/8NBeG/8AoF61/wCAT/4Uf8NBeG/+gXrX/gE/+FehR63pMt2bRLu2+0qu5od671GM9KzofGmhTXF2ou7Zbe0KK9y0i+WWYHgH1GKAOO/4aD8Nj/mF61/4BP8A4Uh/aD8NHrpetf8AgE/+Fd8niPRX1CKyW9tmnmh86NQR86+oNWLbV9LvbaW4tLm2mihJEjxkEIR1ye1AHy3441/QvFHxi0nxMul6udNQRtfI1q2SU+7gY6cLmvXV/aB8MooA0rWcAY4sn/wr0Bte0T9yTqNn/pB2xYcfOc449eanuNW061tJLm4ngSKINvYkYG3r+VAHnX/DQnhr/oF61/4BP/hSf8NA+Gj/AMwrWf8AwCf/AArvIvEelzRi4imt2tGgE4ud67dpOP596WTxNoMVrDcyajZJBOSI5DIu1yOoB/GgDgf+GgvDX/QK1r/wBf8AwpR+0F4azxpWtZ/68X/wr1VVjZQVVSD7Uvlp/cX8qAPHNV+M3gvXrH7Hq2hatcWxYMUks3ABHIOe2K9W0ewsbDSbeDS7dbe1CZjRewPP9ap+Lo0/4QrWztHGnzkcd/LatLTP+QRZ/wDXBP8A0EUAINNtBqBvhAv2kx+WZe+3OcfnVkop6inYooAaUU9RXmPij4r+FtD8WyadqOnX11faUwIeG3aQRl0zxj2JFeoV5h4JVW+N3xEDAEBrLGR/0xoArf8ADQfhv/oF61/4BP8A4Uf8NBeG/wDoF61/4BP/AIV22o+JrXTfFGmaLNau0moLIVlUDbHsBPzfXGKr2PjfRrn7c1y8dnFZ3ZtfNnKqsrAZJU/56UkByP8Aw0F4b/6Betf+AT/4Uf8ADQnhr/oF61/4BP8A4V3765YQtK1xJbxW0cccn2hpF2kPnHfI6daaPEuhte2tql7atLeRtJAAwPmKCAcfmKbA4L/hoPw0f+YXrX/gE/8AhXm3xP8AGHhvx1rvh2/i0vVgLG6X7WTaMC0GckDjk5r6EXxHplxbiXTZ7a9Xz0hbyZE+UswHOT79OtWbTVdKvriWC0uraeaH/WRxuCU5xyKAPOI/j/4YijWNNJ1lVUBQBYsAAPwp5/aD8N99L1r/AMAn/wAK7+XxDosMUksmoWapHII2YuMBz2PvVpL+yaMuJoCgCncCP4un50Aebf8ADQvhr/oF61/4BP8A4Uf8NC+G/wDoF61/4BP/AIV3MXirQZ9PN6uoWa24kMRkdgAGBxirl5q+l6fBHNe3NvbxS/ceRgob6GgDzr/hoTw3/wBAvWv/AACf/Ckb9oTw0qlm03WVUDJJsn4H5V348SaD9lkuf7SsvIjYK8vmLtUnoM1neK9f0WLwjf8AmahZx/abGbyC0i/vMoQCPXmgDU8Oa/beJfDtnrNiH+zXib494wcZx/StbNcP8Gufg74b/wCvT/2Y13OKACiiigAooooAKKKKACiiigArnPFv+u8P/wDYYh/9BeujrnPFv+u0D/sMQ/8AoL0AdAw+X+dcBcRajLqn9lf2bcqlvqEt4b1l/cmNlY4Ddz82MV6A2Mc9Kx59e0+K7t7aYzGW6keNE8h2zt6ngcL/ALXQ8c80AeZ6XY3l1oMtzY6ZqoieCP7aJN+69PmDJi59PTFUtU0DVR4bjh0zRL2Lzbq6lgWSKWWSFWcFMhZFxwerFq9i0y4s7izxpwVYYJHh2om0KVOCAPr6VYkljjZRLIqF2wu5sbjydo9e/wCVAHm2uaNeQyeIH0zTbpLy5WGT7Th3SRABvQKHGW4PAx161a0vRtRuPhxFp05vILh76JgRbmGSFPOUkhXZzhcE8k136yxyq/kyK21ipKkHaR1FPON3bpx9KAPLNR0K9gs4bfUdO1LUFh89I3tcgtMxykz7SPbnpkdKtC21hdZ/s28sr2dJrlJnuQCIQn2QIVLZ67h0r0d2WJWkkdUVVyWYgAAepqCK/t5r57WOQtOkayEDptbofQ9DQB5DqGhaouhaPaadpN9a29urpNEbaWZxKCAJAFlUgEbsEkjnpXWeMfDl1rccQk+2SGHS59jQyGPNx8m04B+9wcfWu7znjP5nPNIGGPXmgDy7xB4bvLLR5rHTbC5uIGvQ6ZSWcplBkgLIhOT3zgelUzaXs/iCws5rHUJ9UgsbBkmQssVswlctvG7AyB3znFevmOmC2iS4eeOJVlkUK7gYLAZwCfxP50AYdgCPiFquev8AZtp/6HNXSVzllx8RNW/7B1p/6Mmro6ACiiigAoPSig9KAOa8J/8AH74k/wCww/8A6KirO8RaLrd3ql1Bp8FtJY6msS3Ezy7Xg2ei4+fitHwp/wAfniT/ALDD/wDoqKrcnifRYtQurKbVbSG4s0DzRyzIpjU9zk8D3NIDhrPw/qWp6pqUcVlbW0EV3Pt1AS/vZCVACldoIX3yajfwf4hl0y88rTrKzmuGt0NvbToNyopBYO0ZUE5B+6a7S48YaHbfY5X1OzNpeMyx3YuU8kMozjdnGTUo8S6ct5fxXFxFBDZrE73E0yrGRICRyTjoO/rTA5Sz8Iava6fY2/2e3eX+zpLSe5aQGSFiSRtIX5uTjIx0p3hnwfqmn+GtZsb1ZI7i8iMaGS6jlDHZtz8kaY/HNdlLqkIgtriBo5obiREjlSVApz0IJI3fQdazdW8baJpej3l6upWd19kKq8cV1HkMTgKSTgfjj9KARg6j4OvhcWqafpthJFJaRWzyPIENoVYEsgx8x69Mc+tPj8Na0bqexltrZtPAuGS5afLSGUcAx44GfeupsvEWl3TeR9vtEu1i8yS1+1I0ka4zkgHgY705/E2gxpE763pypKT5bG6QByOuOeaAOTl8Ialf6fGk9naWx+xQ27wxSZQlJdx5AA5Uelcz430+HRtSsYbiKMW73U00Vukqx71ZlIQZR8nj7oC/WvUZPE+gRWsN1LrmnJbzkrFK13GEkI6hTnBI9qgk8SaUbu5tra9t57q1ZBLAsyKw3EAHk9Of6UAbcJHkJhcDA4p9Z413SjdT2q6nZm4t1LTQidS8QHUsucgD3qWw1Sw1WFptNvbe8iU7TJbzLIoPplSeaAKXi7/kSdc/7B1x/wCi2q/pf/IIs/8Argn/AKCKoeLv+RJ1z/sHXH/otqv6X/yCLP8A64J/6CKALVFFFABXmPgj/kuHxF/3rL/0TXp1eY+CD/xfD4i/71l/6KoA3/FnhzUNWuDcaa8cc8doywSOT8svmow49CFIP1rA1jwPrJ0mWz01Y5fNuInciRI3YCEKzZdHHL89M4PWuy1LxHpmk6vBY6ndwWnnxl45J5lRSQQNuSRyc9K0VvbWTAjuIWLMUXEgO5h1H1GOntQBwtn4Hv1s7WC98qZY4NOjlV3zvMG7zM8DrkfWr0Xhm9sdUs57Kzs5ERrlSXIHkq7BkKjHbHTitn/hKtL/AOEg/seO8t5Z1ieSby5lPlbCoO4A/L97vSjxJp08lj9guYb6G9laFJrWYSICFLHLA46CgDhtD8C67BbyterGkj3tlOAbhHG2KUu5GyNMZB4Byfeuj8PeFX0i4tpTHCku24E7Rnly8jMuT34Nak/i7w/bWzXL61YNCkywO6XKFUdjgKSDwc+tSf8ACRabbpNJfX1raRJN5SyTXUYV/lVgQc8ZB6HB796AOUn8GX9ppNj9m0+yvrmG4uHmgml2I4kZ8MWIOSAwpYPC+vWVzBbxwWs9rP8AZ3uJmn2mLy85VVx8w54rsn17SEvYbR9VsluZwDFCbhA7g9Cq5yQahu/FGiWTXMU2rWKz2sZeW3+0oHUAZ5XORQBwus+Ctau9JitrW0hhFveyusdvcJF56PnDMWjcA89Nv4iujuPCjXenaVBcQRyi0tpIylw4ch2TAOcAHnvj8qq+H/ibo+u3EiqYbaG3t1uJriS9hKRhjwCQ3X+XQ107a9pEUdu8mq2SJdDNuxuUAm/3efm/CgDjNT8GX8WhzWuiW9ujSR26PGjom4r987mRgD7kGuV1nw7q9josmmPptreyxaDIs8txOv7j52bKkL8xx6Ba9Tg8W+H7nT/tyazYC23mIyPcqqh842kk9fasvxv4n0zTfDl7bPdwPd3VpIkVusy+YwaM4YLnJHuBQBB8Gf8AkjvhzP8Az6dP+BGu4rhvgzx8HfDYH/Pp/wCzNXc0AFFFFABRRRQAUUUUAFFFFABXOeLf9boH/YYh/wDQXro65zxd/rdA/wCwxD/6C9AHQMeK4HTvDcj+MLTVL60kL2896UlY/dVnO3jPcE4rvmOBx1qMFc46YoA8qh0TUIdW1VtL0+7t9QZ7tpLuYARTK7koqYbk9OwqnpPh+4bRrd9T067nSDU45DbrbmJlGxgzKPMbPODnI+leuSXUaXMdud29wSMDIwMd+3UU5pUEwjYqHZSypnkgY5H5igDyfWPD93baLqdrpOlXEatqrS3P7syCeM7yu1d65xnnkVbi8N6nPpOoTSJdvcxeH0is2K+WRN++zhdxw2Cnc9q9LtLmK8to7qDcUlUMpZcHB7YqcnOcHrQBwsPhf7J4Z8RaTHbzNazRZhRnJ3uYVyBz3YfnWavhUXM8Oo22n3Vu9raWq2ySNtKMJj5g6n+HPXsa9KyGzgilOO5zQB49baRrDeI/Ejw6dPbRz2tyg2xFd7mVdhD7zuyAT0WvQvCujjQ2vrWFJI7bzEaIO5b+H5jk+9dEFBHHFLtFAC0UuKMUAc7Z/wDJRtW/7B1p/wCjJq6Kuds/+Sjat/2DbT/0ZNXRUAFFFFABQelFB6UAc14U/wCP3xJ/2GH/APRUVZmuaFqsmo6w+mWVrIL62RUneTa6uuAVwVPB9fpxWp4T5vvEn/YYf/0VFXR7BSA8rs/CGvWeh2Xm6dbX88M87tb3F0CFWRcA7gmCR16Cn6p4F1qTTZktJQX/ANFYKs/ll/LVgwyVIA5HY16hsHGeaNg/XNMDibfw3fx+FtFsjHsmtLxJpUkn37VBOcMFGevoKzovAU8elW8AtrUSCFBNk5DsJt+Txzx3r0fYAc0eWP1zQB5rceENUkmurVLG1jt2E7JerL+8l3rgIV2/KAcdzV7UPAovLXy/stsdtlBAAeMFZA7duM4PPeu82CjaKAPMfE3gvWJ7A2miwxLHLczOVS4EW0MflPKN+IGPrWjP4a1J7i/SG1tQt0ls32jzPn3Rldykbenyk5713uwUbBk+9AHmd14L1e5a/tFtLWOJkutl6JiZLgyghVZccAH3NdboGiHRp7vy4444ZRGUSPpkKFJx+FdBsGKAuKAMfxd/yJOuf9g64/8ARTVf0v8A5BFn/wBcE/8AQRVDxf8A8iTrn/YOuP8A0W1X9L/5BFn/ANcE/wDQRQBaooooAK8x8Ef8lw+In+9Zf+iq9OrzHwR/yXD4i/Wy/wDRNAGx4u0S/wBQ1aG4sNOtb1fsr2zfaJdnlbmUh1+U5xjPaodC0Ga28b3bGRjb21quwEcec6qrEfghP/AzXV3up2WnNGL+7gtjIdqCRwu4+2aRdU07+0TZC8g+2AbjCHG/GM5x16UAee3HgfXLqa+jjhtbVRbXUEN4kxMk5lkDKzDbxgD1NWtB8JavbW9sLpGhlju5JX33ImwGiZAQQi9z0rs117Sjbzz/ANo23k27bJn80YjYnGCex4q9BNFdQLLA6yxSLuV1OQwPoaAPLtH8Da5aaTcJd2sVxcpLbvGtxeCRJRG+T0jG3IBHetmTwbeXWsvcXdvbNbtfS3Ii37toa1WNeMdnX+td7gd+tLsH0HpQB5FbQf2f8TNNtrmLzZIEijEay4CnYfm2lPmAz13D6VqXvhPW7/xvd3s8Uf2FkkVSZxtcNHgApsz177jXo/2aIy+YY1L/AN4qM/nUm0UAeV3vw91NdLkXTYoIZ/LswBE4QyGNSHG7acckc4PStDRfDuq6VNpryaZDeqsZimW4utzW3zZ3K2zDfTAr0TaKNgzQB5RqvgnX7nR7e2tbWGL7Ncy5S3uhH56P0YkocEemD9ai8TeFdXTR7tFsbeS3W1WVruW4zLF5cTZQDb831yPpXrewc1l+JlA8I6uR2sZsf9+zQBzvwb4+Dvhz/r1/9mNdxXDfBjn4N+G/+vT/ANmNdzQAUUUUAFFFFABRRRQAUUUUAFc54t/1ugf9hiH/ANBeujri/iVqraJo+m6mLG6vls9TileG0TdIy7WHA/GgDsWAx9eK8nttO1oeM9Xmvbe8KyRzqDHBJtlUjCAPnbnp0ob492hGP+EM8Tf+Ai//ABVMPx4syR/xRnifj/p1X/4qgDStLDVYoIv7RhvzILW7W5kQEl2bZs249gQKoaHozef4eur3RdQX7O9xFnbIpQkqVdlJyF4PWmH472h/5kzxN/4CL/8AFUn/AAve17+DPEx/7dF/+KoAsaX4c1OezsbnUYr1bqxsLLysyMMSB2MmRnk4AzmmeDv33jDWpTJP9qmjk+yk79h5OcEnaeo6VF/wvi17+DPE5/7dF/8AiqoWXxZ0LT7ya9sfh/4ggnnJ8yRbNQW/8eoAuQaHqFzYzW1pY6rbLI1ql6Zi6mZ/PUyMpJ6bPSm+JNC1qCxsLKyt72S0t5Zwm2KWV1/eEo3ynOMdzxTrf9oDTruETWfhPxDPEwOHjtlIyCQeje1Sf8L4tDn/AIozxOc9c2q//FUAerWIcWMAlyXEa7iRg5wM8VPXkg+PdoP+ZL8Tf+Aq/wDxVL/wvy0/6EvxN/4Cr/8AFUAet0V5L/wv20/6EvxN/wCAq/8AxVH/AAv20/6EvxN/4Cr/APFUAdxZ/wDJRtW/7Btp/wCjJ66KvPvAXikeMPE+satHpV/pkQtbeAJfRhGYq0rEjBPHzCvQAaAFooooAKD0ooPSgDm/CX/H94k/7DEn/oqKukryOf4mx+CvE+v6fc+G9bv/ADdQM6T2duGjZWjjHBJH9007/hftp/0Jfib/AMBV/wDiqAPWqK8l/wCF+2n/AEJfib/wFX/4qj/hftp/0Jfib/wFX/4qgD1qjNeS/wDC/bT/AKEvxN/4Cr/8VSf8L8tP+hM8Tf8AgKv/AMVQB63mkryRvj9Yqyo3g/xIrvnYhtly2OuPmp3/AAvu0/6EzxN/4Cr/APFUAetUV5L/AML8tP8AoS/E3/gKv/xVH/C/bT/oS/E3/gKv/wAVQB61RXkv/C/bT/oS/E3/AICr/wDFUf8AC/bT/oS/E3/gIv8A8VQB6D4v/wCRJ1z/ALB9x/6Kar+mf8giz/64J/6CK8i1n41wavoV/p0Hg/xIkl3bSQKz2owpdCoJ56c167poK6TaBhhhAgIPb5RQBaooooAK8x8Ef8lw+Iv+9Zf+ia9OrwyTxi/gP4y+Mrm70DVtQh1JrbypLKDco2RAHJJHrQB3/i7Q7+/1OK4sdOt9QVrdrdhM6jyCTneM9/pVQeDJ1uHuFWIXP2pWFz/H5YttnXr97nFYB+PNmevgvxN/4Cr/APFUf8L5tP8AoS/E3/gKv/xVABpfgjXbLw/fW9xZNc3MgiRM3EWHK/x9MZ7816NoEFxbaJZxXqok6RqsipjAYDGOOK86/wCF82gP/ImeJv8AwFX/AOKpP+F82m7P/CGeJv8AwFX/AOKoA9dwKXFeS/8AC/bT/oS/E3/gKv8A8VUVx+0Pplq8SXPhPxDE0z7Ig9ug3t6D5uTQB6/RXkv/AAvy0/6EvxN/4Cr/APFUf8L9tP8AoS/E3/gKv/xVAHrVFeS/8L9tP+hL8Tf+Aq//ABVH/C/bT/oS/E3/AICr/wDFUAetVl+J/wDkUdY/68Z//RZrzn/hflp/0Jfib/wFX/4qqesfHCDUdDvrKHwb4kWS5t5IVLWq4BZSB/F70Add8GP+SN+G/wDr0/8AZjXcVxXwhhls/hL4etrqJ4pktcMjqQQdx7V2maAFooooAKKKKACiiigAooooAKYy8in0UANApcUtFABijHFFFACYqKXARgehHPXmpTXC/FzW9f8ADvgK61fwu0YuLMh5t8W/910Yge3B+gNAFb4IHPwl03Pea5zz/wBN3r0MV84fs6eJ/FGsXx0ZpYl0LTUeWQLEu4u7Ehc9R8xJ49K+jlPyigB2KTFLRQAUUUUARlTuyOtPHalooAKKKKACiiigCMJhu9SUUUAFFFFABTTx1p1NI64NAHNaxkeOvDeD2u+B/wBc1rpQe1fLHxE+Jvj3w18T5dNNxBJLp8jCxJtVYukqjBwM5JGABX0h4ZfUZPDGnya4yNqMkCtcFF2LuIyeMDHWgDZooooAKCM0UUAN2UtLRQAUUUUAFMC8nPFPooAMUYoooATFLiiigBCOK8x+K4/4qHwB3/4qKIHn/ZNenMffFfK/xv8AG/ivTfihBYSeWlvpt1Hf6WvlKxb5doY46/MG4PpQB9TK+eBzT657wZLq1z4P0y68SPG2pzwiScxJtXLEsBjsQuAR65roc0AFFFFABSEZHWlooAjVdvTtT6WigAooooAKKKKACiiigAooooAKKKKACiiigAoNMZiM4qnqer2ekWEl7qVzHbW8aks8jADpnH1wDQBaZ+TjqK8z8b/EjTJ5J/B2h2n9u6vqUTWjQK37mLzAyHzG9ByTjtWNB4t8W/Fa+W28JxT6B4eYN52pun72Q4UgJnp35HrXofhTwBoHg+3ZdIslE0hzLcSHdLIck8see5oA8g+Gc2ofBmW40nxtpQtrLUZlli1G3zIsbH5djnsAFz7c+te72mo22oWiXNlOk9vIoKPGwIIIyDTr/TbTU7OW0v4EngmVkdHXIIIwf5mvI7rwb4g+Fl9caz4HuJtQ0NFLz6LO5bA2gEoeuQRu+nFAHs6ninVx3gj4jaN45tpP7MkaK8gz9os5RtkiwxXkfh+tddu9aAH0UlLQAUUUUAFFFFABRRRQAUUUUAFFGai83FAEhbHWsnXPEWm+HNPkvtXukt4YwT8x5PsB3JrkfGPxUstNu/7C8OL/AGt4guEYQW9v8wRuxbtjr+VZnhj4aajqWsDxH8Sbwarf4zDZf8sLY5zkL6g/kaAPOfEcfiT4g+LLPx/p3hHOl6U0ey2nfE12qtnd0568emO+K9p8D/EfSvG8Ei2qSWd9b4FxZT8PG3oP73pmuls7i1mR1syuyB2hYIMBWHUVyXi/4bWGutLqeks2la7GpaC/tTsYtjA3Y6jHH40Ad3u//XS5ryPw78Rtb8N6mug/FGBLRwNtvqqriK5O7Cj0GR/ImvVobhJ4VlgdZI2XKupyG9we9AE9FGaTNAC0UUUAFFFFABRRRQAUUUUAFB6Uwsd1c54x8c6P4J0n7brVwE3hhDEPvysBnCj8vzoA3ZbmOCJpJpFjRBlmY4AGM8189fFC7tfH/i/TLvwbpc2uTaG5N5Kg/cyKo8wR7upJxx78V08Wl+JPjDBHNrJl0LwuZgy2S5We5Cv1Y9QCOPqK9P8AD/hfSfC2mLYaFZx2kAwWCDl2wBuJ7nAHNAGD4I+IemeMLP5A2n6lGWWfTp2/eRbTt9siuyDZIrh/G3w1svEyi90yT+ydciYNDqFsNrjGflPqOa5zQPiFq3gqeDQvioPKkY7LXVUX9zMoUdT6/wA6APXxRVaG8juYVmt3WWJ/uujAq341YByKAFooooAKKKKACiiigAooooAKKKKACiiigAozRTSwHWgB1NMgBOfSq99qNrptjNeX8ywW8CGSSRzgKo6mvK5/iHrHxE1SfRvhtCILNFIudbukOwLnBEajqcHIOe3SgDpPGnxL0zw9v03Ti2p6/Ip+zafbLvctgH5sfdGDnmua8O/D7XvE+qnXPilKk4B32mkxvmK3O7KlscEgcD6nNdJ4J+Fmj+Cz9rhaS+1WRcT6hccyMTjO3+6Dj39M12qphv1zQA23s4rWCOG3RYoo12oiDAUegFZV14osdP1m7sdQkW2S1tYrh7iRgFxI7oB9cp+oraVht61xniTw1qGoaxqV5ZwQXC3Vjb20cctwYsMkkrFshWxw47HpQB1trdQX1slxaSLLDIu5HU5DCpTHn0rO8PWl3ZaDaQagyNcRx4fYcgH64GfrgVpFlHU0AcJ4t+GVjrF2dW0KZtH11Msl3bfL5h5O1wOoLEE59KwPDHxN1fQ9Qh0D4o2P9mXJi/dalkGCfaq/xepzn68V6ueTxxxzVHV9B03XrF7PWLKG7tm6pIuf89B+VAF5biN13IwYeoPFShsgV4rHo3jH4S6gbjTLmbxH4U3b57Rxm4tUG4naf4sAj646CvSPCfjbQvGOlreaJepMOjxk4dDgEgj2yOnFAHR5opm8U8HIBoAKKKKACiikLAUALTTIAaTzF9/yriPG/wASNP8ACUbW9uo1LWZCPs2mxP8ANIc45IztH1oA6zVNYsdG0+a+1O4S2t4VLO7nAAAzXlI8Y+I/ihfvY+B4ptI0Vc+bq9zEVM2Djag68j8qdo3gDxP431SPW/idcrHaI++DQYgfLHBH7w98Z6d/avV7awhsbeO3s4o4YYwFVEGABQBzng74caD4KtsaZbiS6YfvbuYZkkPc57V1YTHI/GlLAcGk3jb14oA8t1y1uYZU/wBFvpJP7ZllSKGOTY6lgASy8D/gXFenou+FQw25XkZ5FRQ3UVxvMT52SGNjjGGHUc0+C7huN/lNu2OY2PT5gcEUAVdT0Oy1myks9UgjubaQcxyICAcEZHoeTzXkjeHfFXwjuHvvD9xPrvhiGMtcae7/AL2BQBgpnt7DsK9rBz0qN4t2QVDAjBBHWgDA8KePfD/jOzM+h3okZCBJDINkkZ56qeex/Kuh3jng5Fed+JvhJaXWrf8ACReE7j+xtdhfzkaMfuZ3GTtkX3JxkdB2qn4P+Kty2rx+HPiBpraLrXIWQn9zPhQ2QexOTxz9aAPUwaWoxKrKCp3A9CKeCMUALRRRQAUUUm4UAKTiomnRMljgDqfSs3xB4n0jwxpcl/rd7HawIM5c8nkDgdTyRXl93ceKPjFHEuitL4c8LP8AMb1xme8UqR8qjgDdnv6H2IBr+LfifcrrSeHvAVh/bWsnIldD+5tjuCkSN0Hf6VJ4T+GP76HW/HU7azrbopZZyGit2IO4IvTkbQf92uq8LeDtJ8IaWLHRoGVQzM80p3SyEnJLN3reA29qAEEXzE5+g9KlxRRQA0pn3rL13w5p3iPTZbDWbWO6t5ARtdeme4PY+4rWpCR0zQB43caV4n+E6i48NLLrnhsSkPpmMzW4JJLJ6jtj3rvfBvj7RPG+km80acloyEmgkXbJG2M4IPP4966JlDVwHiz4YLeSSax4OuDoniBMsksRxFOeMiRfoMZHTNAHoYkBp2a8s8J/Ey9g1P8AsD4j2SaLqy7Vgn3ERXeSRkf3emcEmvT0lSRd6HcvYjvQBJRSbxx70uaACiiigAooooAKKKKADNAOaztdleDQNQliYo8dtIysOqkIcGvLNP8Ahx45vtLtbz/haGqR/aIUl2CPO3coOPve9AHslcj468faR4JsVe8b7TfzFVttOhYefcFjgbV69e+K5L/hVvjn/oqeqf8Afr/7KuVTQdS+DnjC68VeIoZfFdlekJLqMY3XFqeSzbPfpnPbtQBuaX4V8X/E67S/+IrPpvh8ktHoMLNEZQRwZCCG69VJyCO1es6dpVnpOnx2Om2sNpbRjCRQoFVc9cAeuTzVTw94l0nxTpa6hod4t3bOdu5f4W7qR2I4rYGCo5oAUcL+FYEni3Sj4mXRYb23nulSRp445lLW4XbneoOR171vEenevNZPBGuXN1dQuLKC2CXiwXMUreY5mcMCw2/LjHqaAOv/AOEx8N/2YdTOvaaLHzPLNz9qTy9/93dnr7VEfF+jTXc9lZXttdXUEscUsKzqrLvYAHk+/wCPIrj4fBniSDw7PCsMLag9wJFkk1RyUAXGQ3l8em3GMVsXHhrWZrm5R1sngnurS5M+4q5MboXG3HTCEjnqe1AHRJ4m0N5rmFdXsnltEMlwguFJhUdSwzwB71Ubxz4ZENrOmt2EkF1OYIpY7hGVnAzjOcdqy18J30FiTZG3S8xdkSbyPmlYlOcZ9PpWTo/g3xBp9pHNOLaa8j1Nboxz3bSgJ5PlnDlAc8k4xQB13/CWaRb24l1S/s7ASXEkERmukxKyMVIBz1yPu9R6VdHiDSf7XGlDUrQ6gRkWvnL5hH+7nNcQ3gzXLWN3s4LC5knkvVdZ5mURJNcGRWGFOSAenHPetmw8JTWV0JWMTut8kxkydzIsWznjqTzigDq2UMpwAcjGK8y8SfB63k1Jde8DXLeHNbjdXzbsRBcYKna8Y424XGAADnnNeoZ+XJxUMs6QxvI8ioiAszucBQPU9hQB5r4U+LP2nXIvDHjjTpdC8QuSqK6MsVzy2DGx7HaAOeTwK9SVsqPpXg/xC8Rr8Srq48K+DNAXWLmGTY+rTqFhtGyAXRu+AxGRyDyM16v4J0W/8PeCtL0rV74315aw7JJz3OSQAe4AO0ewoA6KigdKKACql9eW1jaS3V9NHBbRLukmkbaqL6kntVuvOvjD4O1rxj4RWy8P3SRSwzieS3lJCXIHO1iPzxjnpQBjXvxC1bx/eXeg/DCIGFMxXetyghIM5GUz95uh78HPuOg8CfC7SPBsn9ou02p67NHifUruRnkbIGQMnAHGM4zjg1kfDfxtptvbR+GNZ0qLwrq1qfLSxcBY5RjcWRunU9M969PVwVyO9AEoGBRSg5pKAMzWxcnRLwWTMLkwv5YXOc47e/pXnmm6bc30MVvZJr0OnSXUK3AvJLhJmOP3hDOd4Q4HKkD6V6kwB4P16UmMtjtjGD2oA8i8WLeh4rWWXUo4Uv5mZo2uWdYcjDfuiGC4BwzfLxWmdIvdWRIpptVFqr3csbRzyxmQAAxZYEEj0BOD6V2HiHRtB1GOK58QWkEy2jgo8y7ghJABH4kVqSzQWVk08r7IIk3FlUnAHsOfSgBNJ83+ybX7TkS+Uu/d1zgdau1maVrNjrtr9o0y482LcVJ2MpBHbDAGtGgAIypFYniHwpovizTzYeINPivbfOVDjDJyD8rD5hnAzgitzOKMjNAHiyN44+E+oRpdPN4k8HhiTIFMlzYx5c5JHzMFGCScgDgYr1Dw74l0jxPpSahod9Fe25AyY2BMZIB2sB904PQ81k+M/iD4f8HWh/ti5864kIWKxt18yaUkgYC9vvA844HevP8A4d/DzWY/iE/jU2qeGNOnBCaLE24su3ad+MAbmUP360Ae4UUUGgBCa848cfFez8NaodA0Oxn1vxHNGWisbZSxQlSQXxzgYyR1289Oa3/Huna7q3hG8tPCl4tpqj7fJlZygHzAsMgE8jivPvhLrHhrw0U8OavYNoninYDdPeAZuz8zblk7jbnrjGcc4oAv+H/hpe+J72HxF8VmN/qMbMbXTAwFtZrvyowvDng5yTkHnNeowW8dvAkMMaxxxqEREUAKAMAADoAOKlRgwznOec0/3oAwdc8Wab4duYIdRS7Zp0eTNvbPMEVSoZmKg7RlhycVuJIrqCrbge9cT418PX+tatBNYvMggsZwQj4SZi8eI3HcHB/Kufh0PWZPiJJfT2VxErg+VMtvGQsZTiMyb93XPG00AerlxS7sda8ptvh4llPb3Nnpzx3MSWbq+4cSh2MrfXAGabZeHb9Jbw2Wk3FnerHI1xdyFdt5JvLJtIOTggcsB0oA9XLjB7YqrcX8Frc20ExKyXTlIxjqQMn9K8l13w34guxpVxdWlxKX3yyrFCkzW8jOCD80i7cLxkZ6Vt2WhXP/AAsGC7n0e5+1w3LSTaq+0JJF5eAow2cg9sfjQB2Ft4jsrvXpNKWO8iuYgSDPavHHLjGdjkYfr2ra6j/GuftdMMvjC71OeW7cRwpFDHIR5K55ZkGevTPAroR0oA53xV4Q0jxno7abr1mLiDO9CSQ0bdmUgjn+fevNVHi34NzSz3t1ceI/ByY3NI264s8kAEn7zKPbgDpivbDwKyta1bTdG0ua/wBZuo7ayhAMsknQD+Z+lAFbw14q0XxbpaahoF9FeQkjdsb5kPoy9VPsa3c186+HYbnxF8S/7V+FGn3Wg6WzkXuoTACC4wCf9SPvZbkHP5dK7/xlc+NfCXg3Udc/4SGxuRZRb/J/s4Lv+YDru96APTc0VQ0i4kvNJsrmY/vJrdJHx0yVBNX6ACiiigAooooAzfEX/Ir6p/15y/8AoBrJKag/w7s/7HuDBeJYRPGwUNuIQHbg+ta3iL/kV9V/685v/QDUegRmTwrpYBx/ocXI7fIKAOcs/GDrp8eqzvJOl/E0ttaDClVjUbyOMkkmtjTtXtPEUmoWhtwUt28qVJcHfkc8ehqBvAmn/wBm6baJPcxnTNwt5lk+cBvvDPocYrR0zw7aaXez3UO9pp0VGd2ydoJIH60AcD4g+HN/4fu5Nf8AhnO1le7w0mlM3+jXOW+bI/hOOOPSr/gb4oxeIriTSdetP7F12Eqr2spwJSRnKZ7V6GY8n8c1yXjb4c6V42slW9DW97CS8F5AdskbYx1HJHTj2oA6sSA9+RTZ5oraF5ZnWONFyzMcAD1ryDT/ABJ4o+F919h+IMkmpaET5dtqyLuZSWO0SHtx+WK9EvZ7Xxb4Puho1zDdR3tqRFIjgg7l4+nBoAtf8JNojWD3v9pW32ZX2GXeNob0zVXVvF+laXBbMbmKSS6MYt4w4zIHYKCPzrF1HwpcwS/aNKsrWZYZ0kW0chY3wm0596yj4G1lXtoltrWaORbYyyu/NsY5S5CD05xx6UAdvJr9o8t1a2MkV3fQIWNqsgDnjofTqKPD+tNrVg00tq1nPHI0csDMG2MD6jrxg1ymg+DtW03xXd3l2zywOs3lubnIO/OBtxkdQPwrr9A0iDRdGgs7eAW6IMsgbdhick578mgDUHamkkZx+lIZFXAz0HevNPEfxXjuNa/4RvwJD/a+tPKYWkQZhtz8wJZunylaAOu8UeNNF8IWP2jXLtYVYkRxry8nQHA79R+decXOi+MPize+dq01x4a8MbMJZxkefPkMCWPbJOMdMe9bnhv4TRHVv+Ej8bT/ANsa5I/nKshzDbMdpARenBXr6V6SsW0Y5PFAGfofh/TvD+mR2Wk2kVvCgHCL944AyfUnFaeKUcCigAooooAKQrmlooA5Hxr8PdG8b2aR6mjxXEHNvdQNtkhPqD/SuFste8SfCOb7L40lfWvDrSCO21NB+8hySf3g78d/WvZivOar3lhb39u1veQRzwyLtaORQykfQ0AVdH12x13Tor7SLmO5tpBkOjZxxnB960wa8i1bwJrPgWS41b4YO6RZ8ybRG+aKUk/wA9CAP1ro/AnxO07xg0lnPDJpmrW+PPsbj5WySR8ufvDIoA6jXhN/YF8LbcJjbv5ez727acYrz650q60rSvs8r6nLp7vBJdeXIzSnK/MFOc9ewr05nBboTTtvHBoA8b1K3vrqys4b+DU5MxxmxCbsDE2T5vvtHetPQDrX/Cxrv7fvjjUSAgxyFWQD5R12fkK9R8rPWlKZ6+uaAOf8KaW2m6Y0s08lzPdyGeV3jCck9No6YGK6Ko8hc7vXisjxJ4u0fwpp5u9Zu0gBOEjz88jYOAB1JOKANaSUKpLEBe+a8n8SfFq/1XVLjw98M9O/tW/VSsl6+fJgPzg89z8oIPSqE9r4y+LGoq0k1x4e8JM2UjA2T3afKee4B5+leraH4Z0rw3YCz0SxhtIV/hjXG7vyep/+vQBx/hT4W2On37a74ndta1+4YSS3NwAVjbjhF6AAqMGvRBGB0pQvvTqACiiigBrIDXKeMvh9oXjW1KatagXKoViu4uJIzg4IPsWJweK62m7aAPHW8UeJfhRcRWXitW1vw9gCLU4kIlt13BVV8cHjnPcmvUNG1uy17SbfUtMuEuLW5jEkbqf5+hHTFXLmziu7doLmNJYnHzI6ghvqK8l8QfD7xD4P1KTXPhbciGAkGbQycwvhTnaD3J5x6nNAHr+M84oC/SuO8J/EnSvEsp0+fdp2swjE+n3PyurDAbGeo3EgEda7DzBnFAC7etG3n0p1GKAG7aMf5NOppbHWgA28HHemlyPzqG6v7eztWuLqVIYV6vIwUD868t1Tx9q3j1rnSPhgjGINsn1luI4uh+QnqSP5UAdT4y+I2m+FI3twft2qyqRa2EA3O744Bx0BNcho/gfxL461Max8TZljssB7fRbckRqcggue/PY811PgX4Y6d4OEl1NK+p6rc4M99c/MxIJ+7n7o57V2vlYOR1xQBwkvid9A1q98P2+nWdtHCkTaeVUiOQMRuQgH7wySMVT+MOr28nwq8Q2LSZuPsxAwOCyuu4D6V0ms+D01trg3cwy1xFPAyrgxFMfzxXH/ABb8NpafDTxNfGXeAkk8akco0rpu59OKAPR/Dw/4prTP+vSL/wBAFaNZ3h//AJFnTP8Ar0i/9AFaNABRRRQAUUUUAZviL/kV9U/685v/AEA0eHP+RX0v/rzh/wDQBR4i/wCRX1T/AK85v/QDR4c/5FfS/wDrzh/9AFAGlRRRQAUGiigCpf6dbanaSWt/BHcQSDDRyKGU8+leV33gLWPhy93rHw2mEloR5tzo0+SJSox8jdjySfoK9fqNxnpQBxXgf4l6R40mntIo57DVLbiewuxtkTkj6Hp26V2ZYBuT3/OuK8Z/DWw8Tst5YSf2PrcTq8OpWy4cEHOGAIDAk9/auZ0Px/rfg65TRPi1GsBJ2WusRKfJmUKM7z/eGeSOPmAxQB64SMDkVn654i0rw7pzXur3kdrCoJy55bg8AdzweK5HxN8UbDS9QXRvD0D69r8wBjsbU5wNoYliPujblhWR4f8AhZqer61B4k+JWpvf3yO0kOmxkfZ7flsAj+LAYHtg8c0AZ09940+K2qSw6EH0DwqGH+mygrPdplTlV6jIyQfTg816V4Y8G6R4S0/7JodmlvG3Lv1dzyeT361sxRLFGqRoqKoCqqjAUDpgelWB0oAaFx0p1FFABRRRQAUUUUAFFFFABRRRQBGY81xXjv4aWPjG3jnhlOm6tbvvgv4BiRTg4BI7c13NB6UAeRaR4213wJINI+J8LyQmTZba3Am6Fl4VQ+OVOfUV6nZahaX9tHPY3EdxDINySRtuDD6iq2raRZa5p82n6papdWkwxJFIMhv8K8sl8Ja/8JRdaj4IlfVtDCmW50m8fMsaqOTE46fTHQUAey7hSGRR1OK4Pw58XPCuv6HPqD38dibVN13Dcna8PIXJHoSeK5P+3vGPxU1A2nhqF9C8LCQrNqhH765QEhvLzwMj2465oA3/ABH8VrYayfD/AILtTr2uMDhImxDFjBy79PXpVfwx8IG/tweJPHV9/bGseaZokBJhtyWJwoPbnp7V1nhPwLoXgyxFtoVisbMP3lxId0khxjLN6nA6YFdOOBQBEIiOABj2qWiigAooooAKKKKACiiigAqIxHOR/OpaKAOJ8a/DHS/GckN3NJJY6nb/AOovrfh05z+PT8M1ytn451n4a/ZNI+JsPmWeBFba1a5kWTaOd6/eB5AHHOCa9grN1TR7HXLGSx1a1ju7aRSrxyjOQRg49OPSgCex1Kz1K1S60+5juYJPuSRNuVvxH0NWRIprxvU/DPin4X3AvPh5E+q6CW8y60e4bcyEsN3lEAYG3jkHGO+a7fwf4+0Xxho63dpcJBcKoNzaysFeBj/CR+HHtigDrd6nvXL+NPHekeC9Je61GXzZsfubSIgyzHIGFH49a5PxH8SL7Vr2Tw/8MYE1TV922W7YE29su7DEkdSDx7ZB5rQ8F/DBNJk/tjxbc/294gmwz3M4+WE7cFUXOMe+OevFAHO2fh7xN8Wbg3fjaKTSPDiOslrpin55+SP3npx29a9U0fQbHQdPjstJtYrW3jxhEX07n1PvWgowTjv1qSgBAKWiigBMVwvxp/5I14k/69R/6MWu7rhPjT/yRrxJ/wBew/8ARi0AdT4f/wCRa0v/AK9Iv/QBWlWd4f8A+Ra0v/r0i/8AQBWjQAUUUUAFFFFAGb4i/wCRX1T/AK85v/QDR4c/5FfS/wDrzh/9AFHiL/kV9U/685v/AEA0eHP+RX0v/rzh/wDQBQBpUUUUAFFFFABRRRQAhx3OKyte0iz1/R7rTNTt1urS5jKSRP0Ix69iDyCMEHkHPTJ8c3erWFrZXuiO7yW90DJag4FxHtYsv14496zoPGkFp4dl1QPJexypd3ihjhhDDgsAMdQDwKAMD4B+HNI074e22qWmnwrf3M04luioMpAlKhQ3ULhB8o4zk+9etKMDjivO/gSyzfB3SZFHDyXBGRjrO9ejUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUZoAKTIpa5/xib8eFrx9Iumtr1FDQuDxuyMA+xoA3waaeTXDaH4xa8W4nuPMW4aeGzNlI2BDOVORn0JHWuo0PVk1rRoNRjjaNZ1LbG4xg4P8AKgDh/EvgLwxqHxK0G+vtFs5ppxcG43RDbOVQEF1xhz8x5IJ6egr0aKNYYljjVURQFVQMAAdgO1fO3ib9oGTTvHMK3vhiSObRpZ4njN0AZNwAz93jpn8a948O6pLrXhyw1K5s2spbqFZWt2bcY89ie9AGsSB1ozXJ67rmpaTr8NqDD9nv4WjtD5eSLgdAeeQRk/8AATVubxVY2Di31CY+akbGSVE+QlSobHpgsOvrQB0WaKpadff2hp8N2kbxrMoYJJgMoPY4q7QAUUUUAFFFFABRRRQAUUUUAFIGBPH8qWuW8Q63f6Jqtk2I3sLrdBjYSyzHHl856E8flQB0zYYfzrxD42+CNF1jXvCkhtIYLnVdXS0ubmFAryIy87iOp4wCckV6d/wkkGmwJDrM265iiLTywp+7DKu5gOcjjnH0rifiNfJqmofDu7gSRI5PEUe0PjOBuGeCfSgDvPDXhfTPCGjw6TodqLa0jYsFXqxPJZj1J7df0repNopaACiiigAooooAK4T40/8AJGvEn/XsP/Ri13dcJ8af+SNeJP8Ar2H/AKMWgDqvD/8AyLWl/wDXpF/6AK0azvD/APyLWl/9ekX/AKAK0aACiiigAooooAzfEX/Ir6p/15zf+gGjw5/yK+l/9ecP/oAo8Rf8ivqn/XnN/wCgGjw5/wAivpf/AF5w/wDoAoA0qKKKACiiigAooooAp3emw3lxbyzb828nmIAxA3YxyO/U1nP4U0ryFiFqFjj83aqtgYkOXH0OBxW7TX+6aAPPvgWAvwh0sDoJbkD/AL/yV6HXnvwN/wCSRaZ/12uf/R8lehUAFFFFABRRRQAUUUUAFFFFABRRRQAVma3rtpoFkLrUC4jaRYlEaFmLMeOBWnXN+MB8uj/9ha2/maAKo+JOh4/1eof+AUn+FVb/AMb+HdSs2trhNUEbEE7LSRT1z1ArtgOKXFAHnk3iTwjNLNI1nqCyTyxzOy2cgJdBhWzjqAa0LXx94es7ZILeHUUjQYVfsUn+Fdnik20AfLnj7QdN8RfGjTdYtbK/OkXTJJqDCybCsnbGOd20Z+pr3OL4h6BDGsccF+kaqFVVspAAB26V15QmlwcUAeeaz4t0LWLrTJi+pw/2fdi6VVsHO8hWXByOmHNZs+qaBLdak63OpCC+O7yW01mEZYgvgkcg46V6rg0YNAHDaT428P6TpVtYwjVHjt4xGrSWchZgPXirv/CydD/55aj/AOAUn+FdbikxQBz2k+NtJ1rVFsLP7QtwyNIFmgZMqMZxn6iujHSua1FSPiDovPJtLr+cVdKOlABRRRQAUUUUAFFFFABWPrehjWobRWnaH7Ndw3QKqDuMbhgOfXFbFJQBzFz4P87U9Ruor540voirQmNWVHKlS/PPTHHTgVxfj/TItEu/hxYW7MyQ+IowpY+u4/1r1uvMfi1/yMXw+/7GOL/0E0AenUtJS0AFFFFABRRRQAVwnxp/5I14k/69h/6MWu7rhPjT/wAka8Sf9ew/9GLQB1Xh/wD5FrS/+vSL/wBAFaNZ3h//AJFrS/8Ar0i/9AFaNABRRRQAUUUUAZviL/kV9U/685v/AEA0eHP+RX0v/rzh/wDQBR4i/wCRX1T/AK85v/QDR4c/5FfS/wDrzh/9AFAGlRRRQAUUUUAFFFFABTX+6adTHPymgDgPgb/ySLTP+u1z/wCj5K9Crzz4Gn/i0Wmf9drn/wBKJK9DoAKKKKACiiigAooooAKKKKACiiigArnfF/3NG/7C9t/6Ea6Kuc8X/c0b/sL23/oRoA6MdKKM8UUAFFFFABRRRQAUUUUAFFFFAHO6l/yULRP+vO7/AJxV0Vc7qP8AyUHRP+vO6/nFXRUAFFFFABRRRQAUUUUAFFFFABXmPxa/5GL4ff8AYxxf+gmvTjXmXxZ/5GL4ff8AYxxf+gmgD02ijNFABRRRQAUUUUAFcJ8af+SNeJP+vYf+jFru64T40/8AJGvEn/XsP/Ri0AdV4f8A+Ra0v/r0i/8AQBWjWd4f/wCRa0v/AK9Iv/QBWjQAUUUUAFFFFAGb4i/5FfVP+vOb/wBANHhz/kV9L/684f8A0AUeIv8AkV9U/wCvOb/0A0eHP+RX0v8A684f/QBQBpUUUUAFFFFABRRQaACuE+LmteIPDvgO61fwtJGtxaMrzBot5MXRsDHUZzn0FdwXIzkVwHxQ8Z6bonh+80nyV1LVtRtzbwacpO6TzMoC2OQuSf8A61AHl/7OPijxPrF+dGaeMaDpkTyyKsS5LyMxC56jkk8f3a+kVPyivAfg9OfhfcXHhfxjYHT7vVLzzLa9HMMmY1xHu7YIx9TXvKShow0Z3LgEEd6AJqKQHjmloAKKKKACiiigAorJ8Ta3/wAI74av9W8g3P2OEy+SrYL47Zrzr/hcPiVSf+LYawR2IuV5/wDHaAPW6K8l/wCFxeJf+iX6z/4Er/8AE0f8Li8S/wDRL9Y/8CV/+JoA9ZPSvnH45ePvGfg/xlBZ29xCumsyXtkzwIcMvUevB9eua7X/AIXF4l/6JfrH/gSv/wATXC/E7Vdc+I1hZQy/DnVrWazuFlEjSqwdP4kPy5wR70Aew/DHVdf1vwBYan4odGvrsNKNkXl7Yz90EetdmK8dtfirr9jaw21t8LdYSKFFjRBcr8qgYA+76CrH/C4vEv8A0S/WP/Alf/iaAPWqK8l/4XF4m/6JfrH/AIEL/wDE0f8AC4vE3/RL9Y/8CF/+JoA9apM1wHgf4lXPi/X9Q0i+8PXOiXVjCkzx3EwdiGPHAAq1498fS+Cm0uO10WbWLjUpmiigglCNlRnuD2oA7aivJf8AhcPiUf8ANL9YOf8Ap4X/AOJo/wCFxeJv+iX6x/4EL/8AE0AetUxuRXk//C4vE3/RL9Y/8CV/+Jo/4XB4lOM/DDWBz1+0L/8AE0AeWeOPij8QPD3xPm0w3ML3VjI0FqfsqEuku0r8oznIC+9fTehNfv4f09tYdXvzAhuCi7VL7RuwPrXzz4lu9V8SfErRfFdz8NNVH9nD9/bGZf3xUExtkLwVYg9+FArvh8YPEo/5pfrH/gSv/wATQB63RXkv/C4vE3f4X6x/4Er/APE0n/C4vEv/AES/WP8AwJX/AOJoA9boryX/AIXF4m/6JfrH/gSv/wATXeeEfEMnijw3barLYyae8xdWtpH3NGUdlIJwO60Abopa474g+Oz4F02xul0qXVJb27FrHBDJsYsVLDnB9MVy3/C4fEw/5pfrH/gSv/xNAHrVFeS/8Li8Tf8ARL9Y/wDAhf8A4mj/AIXF4m/6JfrH/gQv/wATQB6y3SvlT44+OvFek/E2GymaOO10u5jv9LzGhP3cBiep+YNwfSvUv+FxeJf+iX6x/wCBC/8AxNedfEaXWviFrWj6jL8PNYtGsJAJ181G86LOdv3eDnp25oA92+H+paxqvgPSb/xGqpqVzD5koRQAQzEqcDgfLtrqAa8ih+LXiK3hSKL4XauqIAqgXC4AHA/hqT/hcXib/ol+sf8AgSv/AMTQB61RXkv/AAuLxN/0S/WP/Alf/iaP+FxeJv8Aol+sf+BC/wDxNAHrVFeQS/GzVrJrc6r8P9SsYJp44PPmuVCqXYAfw+9euoSetADq4T40/wDJGvEn/XsP/Ri13dcJ8af+SNeJP+vYf+jFoA6rw/8A8i1pf/XpF/6AK0azvD//ACLWl/8AXpF/6AK0aACiiigAooooAzfEX/Ir6p/15zf+gGjw5/yK+l/9ecP/AKAKPEX/ACK+qf8AXnN/6AaPDn/Ir6X/ANecP/oAoA0qKKKACiikJA4JoAWmFgfwpWZVUliAAOSTXlfi/wAd6jq+qHwp8OS1xqi3Aivb9UzDYryTluhPUY9jQBc+IfxGuNIu4PD3g+1Gq+IbsSYijbIgCjBZvQ57H0qXwF8ND4elm1vxLdvrHiG7VfNuZvm8oAcInpznn6VoeAfAFn4J0dUO271SUl7q+cZeR2+98x5xxXZqw29aAOf8WeDdK8aaOdO1qEyRbg6OjbXRgcgqRyK860jxPqfwnvrPw545m+0aJIZFsdW5JQAkrG/vgfy9K9mxWZreiWGvaXLYapax3MEqkFZFztyMZHoeTzQBdguIri2SaB1eKRQyOpyGBGQR7YqcdK8ThXV/gpdTm4+06t4OkMaJKX3S2L9DleuwAH25WvXdL1aw1mwS80u6iurZ8hZIm3KcdRQBfooooAKKKKAOX+I4z8ONb/69WrT17WRommi5FrLdsZFiWGHG5mY4GM1mfEj/AJJvrmP+fVv5irPivRDr2mxWTKTCbqKSTa+w7VbJwfWgC5ba1byx263TLZ3U67ltZ3USD8O9TJqthIziO8gZo87wsgO3HXPpXDah4PvE18Cw05JoHMHlX8sw32oRjkAH5jkHtTLnwLeHQbCC2gijujPKt6VYAvFIWBJb+IgEGgDvLnV9Ps1DXV7bwhjgGSQLknp1NJJqVikhR723R9m8qZBnb6/T3rzTWPB/iS68LQ2qWkdxcm4fzAHj3JGF2xnL8dAPetNvB18dLvt9pDNeva20cRZ1y/lqA657An8KAOpi8WaS9jd3rXUaWlpKY3nLAoT6g+lPtPE+mXmrzadHdRCeMIVBcfvQwJyvr0rlrbwtfSeCtYtJdItrWa9nM0NiHRkTkHbkcdqI/BtxPqj6kdOgtbkyWxiwykwqikMoI6DJ/HFAHcW+o2d1PJFbXUU0kRxIiOCU+oHSrVed+A/C2r6Jrt5NqsThWUqsvmRkSfMTkhee/evRKAPM/D4/4yM8Xf8AYLs/61P4/GPiX8PPfUbj/wBENUPh/wD5ON8X/wDYKsqm+IH/ACUv4d/9hG4/9ENQB6KKQtzQ3euJ07xhKLnWH1cxxQWUhAgRT5yDcQMqeu7gjFAHb5FN3Ddgda5Ob4i6Jb2VtcyC7IuA5WNbZi6CPG8so6AbuaW98aWy69Y6bZJNI09wqPN5J8tQV3Y39N2MHFAHVdcc85p1cTqviLWrfULu7sYLZtL0+4jt51cnzJC23LL2AXf3/umtpPFOmva/aPMZY/3vJQ8eW21v1oA3M0A59q5zTfGel6msJh+0RtNP5CpNCVbdtLdD2IGc0knjnRorqzhLzE3gUo6wkrGGOF3n+HPv6GgDpM/jXPeCOfDjen227/8ASiSq1t4wi1DxdBpNlHMY9speaSEqrlCo+RujcntVrwP/AMi23/X7d/8ApRJQByfxi/4+PBH/AGMtv/J69L3Dv1rzT4x/8fHgf/sZbf8A9Bauy8VXU1l4W1G5tpHjlit2ZXRcsCPQd6ANosAcUblrz618W3WnWMEUFtqOrztG9zO15GLd441IBO1sZ5YY9a008bB7qR/7PcaelyluLwuMFmAI+Xr/ABDmgDrA4JOPWkIyT7da4x/iBC2hnUbSz3FEWSRJpli2bmKgEtx2pbLx6b+1sri109pI5raS5mZZVKxIjlTz/FyO1AHZ7h+NKGB68VxOk+I9S1K51Zry2ewWOySaCMuH4IJD8fyrlbHxLqsvhOe5k1TU3/eWytIbX/SIy4y4VMcgjoaAPYCwoDA8V5db63qd1quhW8mp6kI57d5S9rb7zLiRQDJx8nHB960PA+r3+oeILpLm9urhFVy8dzDsCESEDyz/ABLgdfpQBH8bRt8C2uDx/a9ln/v8K9HX7tedfG7/AJEK1/7DFl/6OFeir92gBa4T40/8ka8Sf9ew/wDRi13dcJ8af+SNeJP+vYf+jFoA6rw//wAi1pf/AF6Rf+gCtGs7w/8A8i1pf/XpF/6AK0aACiiigAooooAzfEX/ACK+qf8AXnN/6AaPDn/Ir6X/ANecP/oAo8Rf8ivqn/XnN/6AaPDn/Ir6X/15w/8AoAoA0qKKSgBagnlSFXklYKqjJYngAVDqOoWmmWUt5qV1FaWsQzJPPIEROwyx4HOK8ivLjV/jVepBpD3ekeDoZZUnv4pAsl8wyoCdynr24PXigB/iDxNrfxL1WXw38Pp1g0pYf9P1h1OGBYApF6kDP6/Wu/8AB/gzS/BOlix0aN9sjmSaeU5eVz3J/wA9K1NB0Sw8O6Ha6VpNusFpaoERR3x3J7knkk1p0AIPuiub1HxXaaVr17aakBFa2tpDcNNkkkyPIgXA/wBz9a6WuP8AEXhe91HU9RvLGa2DXdjDahZxJ8vlvI5bKEEH95xgg5FAHTWN5DqFjFdWrb4pVDoxGMg1brK0OyubDRba1vp/tE8aAM4yQT7Ekn8ya1B0HagClqFjbalaTWl7Cs8Ew2yRuMqwPrXk2p6Xqvwevm1bwwkmoeGby833ulpHmS1DfxRHPIyTwQP4R717KQO4owPSgDD8L+JNP8VaBbatpEu+3uFyAeqHupHYit0V494n8Ha54G1JvE3wtti0W2Z9Q0NpGMMpI3eZGmeG4xhcdFAHWu18C+PtH8e6ObvRpyZISEuIHGHhbAOCPTng9Dg+hoA62ikpe1AHMfEf/km+uf8AXq38xXQXFzDbQmW5lSKNerO2APxrn/iP/wAk31z/AK9W/mKqfEQNJo9miWS6iTqEA+yM6qJfm+7luPz4qQOiOrWCeWWvYFEw/d5kHz84yOefSo77W7OwimMkyvJCAXiVgWAPtXmms+C9eudO06Gz0uOIBpHMcf2Y/Y8sCFBkUkLgH7hHtWxP4Z1KX+3LRdLRLi9DGLWEePzCCAAnPPGO/FUB3L6nYpsMl3CvmNtUFwNx9B61DFrenSi6KXkR+x5Fxhx+7wM815pq3gnxBO2mBbQslshjkFq1sOd+fNzKpIJHUrg9faus07w+8D+ILdtGto0vSzJcgJmfcgGHA5HPrxUgdBHrWmy28E8V9b+XcDMLFxh/Yc802/12w0+R4Li4QXCQmbyc/MUHpXnsvhLU/sNgieGLVitqLZUEsQWzYMP3q9snGfl5roNd8PXNzqkM0Olw3rtpz2j3TlA8bHocnnn29aoDpLTW9Pu9OF9Fdw+TtBdvMGEJGcE9jVyGaO4hWSJw6MMq6nIavM9Y8B30WhtaaFaRwr/o7yW8CwgTFVIbIkVkJz3YGuz8Habc6T4Vs7K7SRJY0+ZZWQke3yAL+QAoA5Lw/wD8nGeLv+wVZVP8QP8Akpfw7/7CNx/6IaoPD/8AycZ4uz/0CrKp/iB/yUv4d/8AYRuP/RDUAehtXC6h4An1f7cdR1YTLOuy3BttphG7d85DZk6Adq7s1xt58StBs7ya3k+3SPE8i5hspHVyhw+0gYOM8+nepAi0z4frY2cMT3kTtHBcQkw22xW80KM4LMeNvrzU0Xgye31mCcatmyinS4Nobbl5AmzO/dwD6YrSfxZpqatHY/6STIoInFu5hUkZ2mTG1Wxzg1UXx5pkmlfb1tdTZDL5USDT5S8pxnKLjLLj+IcVQFLxB4M1O+a8bSdYW1tbmaO5ltJIN3mSIVON+75VbYAeD39aZF4CuZfPS91VTZzRTqlolv8A6ppTuY792WwenArZ/wCEssV1hdNlt9QikcZEstlIsOQhfHmEYBwDxmqMfxC0maCaSOHUyYXVPL/s+XfISMjapGWGPmyO1AGfZfDm40/S44LLVbeC7iuxcR3KWJAGEZMFTIc8N6io3+F6tfabcHUYZHtIYopPOtSxl2HIIw429T/e7V3FheRahYwXdvuMU6LIm5SrBSMjIPIP1q5QByWk+Eriw1yO9l1Tz7a3837LbC32eWJGBOW3Hdyo7Crfgf8A5Fpv+v27/wDSiSuirnvA/wDyLb/9f13/AOlElAHJ/GP/AI+PA/8A2Mtv/Jq9A1KzTUNOntZRlJVKtzjivP8A4x/8fHgf/sZbf/0Fq7fxHqNzpPh+7vrG3+1TwRl0iLBd34mgCprnhWx164iluLi6t2jQxk28gXzEJBKtwcjgelOuvCum3Ok3unusiwXknmsEbBVsAfLxx90VmP4zuoLqKK60OaJI0jN9L56YtS/QY/j/AA6VJqfjJ9P8QPp40yWeC2ELXN0JVVYVkJCnHVvun6UASz+BdJmtL22Q3EP2yZZ3aOQbkZcYC5BAHHoe9WNP8I2Gm28cERldVt3ty0jgkqzbjngc5NYWpfEpNL0z7ddWC28UszQ2xubyOETbSQx3McLjHQ8nNX7fxo1/Ikmm6a91ZLbC4nu1uE2xZBIXHVunb1oAu6P4TstHS5jhnu7n7TH5TtcyhmCDgKOBgCpIvC+nxSQunmAxLEoG7IYR425461Tu/GHk2scltp0tzLPHA0cSSAZMpGFJ7Y9TVKXx2YLzU7eXTCrWMSvsFzG0rs2MLsB3KMn7x44oAvT+CbGW+guoL2/tZIi+BBKFDBm3EEY6ZFaVpoVpZ3EE0O8PCjIhLfeDHJzx61ix+M7uWa3tW0OVdRluDC9r9pU+Xhd24uPlPH8q7AdKAPO/jd/yIVr/ANhiy/8ARwr0Vfu1518bv+RDtv8AsMWX/o4V6Kv3aAFrhPjT/wAka8Sf9ew/9GLXd1wnxp/5I14k/wCvYf8AoxaAOq8P/wDItaX/ANekX/oArRrO8P8A/ItaX/16Rf8AoArRoAKKKKACiiigDL8RH/imdUH/AE5y/wDoBpfDh/4pfSv+vOH/ANAFN8R/8izqmP8An0l/9ANeW6R4j+MMei2SWfhTRHgW3jEbNO+WXaME89cUAezisbxH4l0zwto0+paxcrDDDGW25G6TA+6o7ntivPh4m+NH/Qo6F/4EP/jXL6QjfED4nTxfFVobK90zZHZaNGSsTseTICfvUAbVrZ6r8Z737brCzaZ4PtboPbWE0eJr0qOsg7LnnHPf6n13TdOtNL06GysLdLe2hXbHFGMKo9BT0jjjVVRQqgcKoxU4PFAC9BWPe+I9L06+Fpd3Yjn2GTbsY4UZ5JAwBwetbFc3qnhr+0rrU5fMVRe2a26/LkqQW5PtzQBZ0/xVpGqRpJZXfmK8nkqGjdCXxnG1gD0BqmvjTRZtYbTo7hvOW2F0ZDEwQId2TuxjI2k1l/8ACL+ICv8AaDXWnrqy3CSoqxOIAqqVwRnOfm65qL/hBtRksHtXu7fZc6UbC5ZUYMGG8gqc8D94eDQBsXnjfSYvCt/rdlI1xb2SHcDE6kvjhSCMgZI5xTdM8caXc6a0t7cRxXEMCzXMcaSOIg2CoyUGT8ynGM89KzdA8D3WneFtV0y6ktxNfBgHhMjD7gUE72Y9u1P1PwNLd6LrNrFcRrLfXEM0ZAKgeWka4Ygg4Pl54NAGuvjTQ2jtHjvS4vSVtwIJCWIODkbcrzxzimaD4y07Xb66sYpdl5bzTI0e1uVSQruDYAPbp0zXNW3gXVray0uG3ewtzbztLLLGZmYZbJ25c5yM5DZGa6HS/CrWGpW90ZY8pJduyquM+dIHAz7YxQB0rLlcn06V5n468BXsGozeLvAMr2evx7Glt432R3yIc+Ww6EnGOevA716eOBUTAZO7ucUAcd4C+Ilt4vtpra9gGl61ZymG606VxuVhzlf7wI54rtQeO9ePfGrTtH0vT4/EmnX6aV4nspBNZtEBvujkKUK/xZz17fTNVdK8b/GG58PQXo8K6SyLFlmuHdJG25BZlzwTjNAHovxHb/i3Guf9erf0ro3C8s+MDnnt715k3ia58Y/s+3WvX0McE15ZyM0cWdq4crxk56CvRr+J5rCaOL7zxsBzjnFICFNX0+WCSeO+tmhiOHkEy7V+pzx+lPW/tGhEguYShXeG8wEEZwDn096831HwLqEPh37NpEQhZobXzo4iuZXQnePmBGTnqQataf4U1RLbRS9u22y5nhuJFzKC/wBz5QBgdelMDtU8Q6ZJpMuo2dwt3bxttcwsCQd20j8/WntrVol1cwzyCAWoQySSMFT5hxyTXL2PhuceCdU0b+zktGllcKYnx5qs+7dx0ODj8Kp3Pgp9N+1f2bZy6hameCU2txclvtAUEEZbOOv6UgO3k1XTohEzXtsvnfNGTMoEnbg55rQABHFeP3GhX7apa2B0ZZppLJyqGUbbDMy4xxzgemK9btojDbRRu29kQAt6kDGaYEmwU7aKUUUAeZeH+P2jPF3/AGCrKp/iB/yUv4d/9hG4/wDRDVDoHH7Rni8/9Quzqb4gH/i5Xw8P/URuP/RDUAeiH+tcyfCa7j/pOQTdcbP+exB/TFdMDQBQBwf/AArdE8Vw63DPZmRNpPnWYkcMFAyr546DtUEvw7v5NGvLL+1bTfd3PnMTYDYhwOQu772QOc16Fs9++aXb70Aefy/Dt08VJr32yO4kt4z5aNb/AL0t5JjAEueBznGK5vSfh7da3o81jfWMenJDdC5j+1WMeJH2lTuVXIfhupI+leybfejb70AZug6WujaJaaeoiH2eNY/3MflqcDHC84rUxTAgHSn0AFc74H/5Ft/+v67/APSiSuirnfBB/wCKcf1+23f/AKUSUAcn8Y/+PjwP/wBjLb/yavQ9RsYtS0+ezuN3lzIUbacHBrzz4xc3Hgj/ALGW3/k1em0AcsvgWw+028rXl+4hVFeNrglZ9n3fMH8WKq3fhD7Z4uvNZvJ7lrbyoPLtIZcLM0Zc/OvQ/eH612dJtHagDlpPCtjq+h2cP+m6eYWM0RSTbLEXJJGRkdSe9WP+ERtFv47oXl8CIPIlj887JxgjLjueetdBsGadj3oA5jT/AATYaeSftN7cnzI3U3E5fZsIKqM9FGOlFz4IsL3ULy6vbq+uGuUMYSSclYQSOUH8PSunx70gGaAMLTPClnpksEwmurmeKRpPPuJS7uWG07ievFb4pMUuaAPOfjd/yIdt/wBhiy/9HCvRV+7XnPxtOfAlqD/0F7LH/f4V6MvSgBa4T40/8ka8Sf8AXsP/AEYtd3XCfGn/AJI14k/69h/6MWgDqvD/APyLWl/9ekX/AKAK0azvD/8AyLWl/wDXpF/6AK0aACiiigAooooAzPEQ/wCKZ1Q/9Ocv/oBqjpGq2djoug2dzcLHNdWkYhRurkRqSBV/xF/yK+qf9ec3/oBrk/EFi918M9LntFm+2WkNtLA0KksGwoPTttJoA7Gw1G11K1W5tJRLC/3XAxnnFc1468B2fjTT0AcWWpW0gltb+IYkhYd89xjtVbVZLqLxAlpEt3FbQwrLG8EbESKEbcpxxuzg+ta3g37f/YQ/tKORZ/OfJcn5hnhhnkAjsaAOI8H+OdZ0TXovB/xGQwXmG+yanK2EvACe/TOB+NesRsCgKkEdiKwvFfhPS/GGkTafrFusqyIVjk2/NEx/iU9u1edeH/Eut/DHUv8AhHviDO1zpWxF0/Vo0JTnCiN/Q57n09KAPZh0qKeaO3heWVgiICzMewFJDMk8KSQuHRhkMpyDVLXNOOr6Je6esnlm4hMYf0JoAz/+E48PDS21L+0UFqsvklirD58ZxjGelSP4w0KO4s4G1GLfeqrQAZO4McD6cjvXO+HfA17pqW5v5LeR4rxbhmWV5NyqhX+IcHmmN4I1OGZIra5tBbTQxR3bsG3gI5YFMDHfvQB2D67psVq1w95GsSiRi3tGfn/LFc+fiV4fN5bhL5Dazq487a3DqyjbjHvWTqvhnWrPS7/dJby2MUVwYI4QxmYynJzx2yenNQ+DfD11qGj+ZNCkCx289spkWQM/mFW34cBuoxz+FAHVLr12vi6PTLi3gFrcRl7eWOXc5xj7y9hz1rpBWJZaKlrrDX8kMBmFrHbrMoO/A6j6Vtr90dqABjjvzXE+P/iDY+CLREMcl5qV4ji1tIV3s7AHBIHRc4BNZ/jb4mppGrR+G/C8H9qeJZnRY7YZ2Rhv4mboMY6daf4B8AS6MY9c8UynUPE06sJJnbcsCtz5a+wyenqaAMXw34WivvE0Wv8AxKuYLzxBNaLcW2nyL8tkiNnKD16E++a9JubmC88O3NxavuiltnZGAxuG01zfjTRp9V1WCfTFdNSsbV5rWXadhbcoKMf9oZGPeqV+usPpMcZikt2SwtWXrjOD5yntwKAMHwTZXWofssw22nwNcXMlhMsUK9XbzGwKs/8ACxfiKD8vw1u2HqZOtbHwLH/FltA/3JP/AEa9ehgYoA8m/wCFi/ET/oml1/38o/4WN8Ren/Cs7r/v5XrQooA8k/4WL8RP+iaXY/7aVSv/AIueNdLuLKC++H1xDNfTeTbo0vMr4JwPfANe0V538S/+Ru+H/wD2Hf8A2i9AGKfiF8Qt+5fhldZxjPmDOKk/4WR8Rf8Aomd3/wB/K9ZpaAPJf+FkfEb/AKJld/8Afyj/AIWR8Rv+iZ3f/fyvWqKAPJ/h5F4l1H4neIPEniPw9NoyXllBBGknO4oTnmrfxUtdeTWPCuseHdHl1eXS7yWWSCM44aPbyfxr0w0lAHk4+I/xEA4+Gd3/AN/KP+FkfEX/AKJnd/8AfyvWaUUAeS/8LI+In/RMrv8A7+Uf8LI+Iv8A0TO7/wC/letUUAeLv8YfGqa8mjN8Pbgag9v9pW3835jHu27vpmrw+JHxFx/yTO7/AO/lad3/AMnK2P8A2LT/APpQa9HoA8l/4WT8Rv8AomV5/wB/aP8AhZPxG/6Jld/9/K9aooA8l/4WR8Rf+iZ3f/fyux8AR6gvg6BtXsn0+8mmnmltpDkx75XYD8iK6mmd6APOPjFpus3uneHrvw9pcupz6brEV40EXUqqt/U1nj4j/EX/AKJnd/8Af2vWfSloA8l/4WR8Rf8AomV5/wB/aP8AhZHxGP8AzTK7/wC/letUUAeS/wDCx/iL/wBEzu/+/tMl+JfxChieWX4bXSJGpZiZOgHevXao6zzoN/8A9e0n/oJoA8l0n4u+N9c0yHUdK+HtxdWk4JjlSXhsHb/MGr3/AAsj4i/9Ezu/+/lbXwP/AOSMeH/+ucv/AKOkr0CgDyUfEj4i/wDRMrv/AL+0f8LH+Iv/AETO7/7+161RQB4R4q1nx/41sLPSrvwFd2MIv7ed5t27aqSBq91TOBn0paWgArhPjT/yRrxJ/wBew/8ARi13dcJ8af8AkjXiT/r2H/oxaAOq8P8A/ItaX/16Rf8AoArRrO8P/wDItaX/ANekX/oArRoAKKKKACiiigDN8Rf8ivqn/XnN/wCgGjw4P+KX0v8A684f/QBR4i/5FfVP+vOb/wBANHhz/kV9L/684f8A0AUAaWKMUUUABAPWsbxFoWn+JNHudK1aAT2tyu11JwV9CD2IrZpCoJzigDxG1n1/4LaxHZ38z6p4Kublo4biUlp7MsNwDnuow3b346V69pep22saZBfWE63FrcIJIpYzkOp6Yq1Pbw3EEkNxEkkUilXR1BDKeoIPrXjUmh658GdXn1Tw4t7rPhGVZJbvTN+42XzbgYlJ6fNzjryTQB7XgYowD1FY/h/xJpfibR4dQ0O+hvLeQA5jfJTIztYdQRnocGtigAoAHpS1DI5VSc4C9T2oAHJB4xjFeWeN/iFq15rT+D/h5AL3Vpocz3itlLFWKgMcex/DjrVDxD4y1/4ia5P4X+GrSW1pbvJHfa6yEwkBcMkbY67iOV54yOOT6J4M8E6P4J0GHT9Hto1bYonuSg8y5cfxOwHPJPsM8YFAGf4N8Aaf4S+0XBeS/wBUvHEt3qFzgySvjnHHyrycAf3jnoK7MDIGRRtFLQAYqnqoH9k3nH/LB/8A0E1cqnqv/IIvP+uD/wDoJoA4f4E/8kV0D/rnJ/6NavRK87+BP/JFfD//AFzk/wDRrV6IKACiiigArzv4l/8AI3fD/wD7Dv8A7RevQzwK84+JL58XeAcnAGu5yR0/dMP60Aej0tRBmLEHj8P61IKAFooooAKKKKACiiigAooppODQB51d/wDJytj/ANi0/wD6UGvR682uz/xknYnP/MtNn2/ftXowJwOaAH0UUUAFFFFABRRRQAUUUUAFUta/5AN//wBe0n/oJq7VLWv+QDf/APXtJ/6CaAON+Bv/ACRbw/8A9c5f/R0ld/XAfA3/AJIr4f8A+ucv/o6Su/oAKKKKACiiigArhPjT/wAka8Sf9ew/9GLXd1wnxp/5I14k/wCvYf8AoxaAOq8P/wDItaX/ANekX/oArRrO8P8A/ItaX/16Rf8AoArRoAKKKKACiiigDN8Rf8ivqn/XnN/6AaPDn/Ir6X/15w/+gCjxF/yK+qf9ec3/AKAaPDn/ACK+l/8AXnD/AOgCgDSooooAKKKKAExUUsCyIwcBlYYZSOCKmoIz1oA8c1vwhqXwyvLzxV8PUQ2O1Df6IVO2RVyWdDnhsdsHqfpXe+EfG+keMtM+16VM6OjmOS2nAWWJh1DL+NdC8QbIIBDDBB715n4r+HE9h4jXxj4EZbTWInMtzbEnyr1cZKEdATjGR6/SgD0W91G302xnvL6ZILeBC8kkhwFA5JNeP6nqWu/F/WJtM0Fzp/g22nRbq9YMst5t+YhB/dJx+h9qr2WkeMPjDNBL4xik8P6BAo3WFsxX7Wwk5DZ5x8uPyIr2iw0+20+0jtrKFYYIkCIiDAAAxQBV0Lw9pvh3RotM0W1SztY8kRoO5OST+JrWxxQOlFABRRRQAVT1X/kEXn/XB/8A0E1cqnqv/IIvP+uD/wDoJoA4f4E/8kV8P/8AXOT/ANGtXogrzv4E/wDJFfD/AP1zk/8ARrV6IKACiiigBG6e3evjz4/abqejfFa4uDcTmG+23NsQcBTtCkKM9j9OtfYhrhPiP8MLD4iRWIvLqWynspfMjuIFUvjHTJ7Z5oAn+FukT6L8N9Gtr0YumgWaY7i2Wbnkn2IrsucdK8mX4K6oFGPiJ4iHsJF/wo/4Utqv/RRvEX/f1f8ACgD1nJoya8m/4Urqv/RRvEX/AH9X/Cj/AIUrqv8A0UbxF/39X/CgD1nJpN3PNeTj4LaqDn/hY3iL/v6v+Fcv4S8Ba34k1fxJZz+O9ehXR9RNnGyTAmQbQcnPfmgD6Cz6UmTXk3/CldV/6KN4j/7+r/hR/wAKV1X/AKKN4i/7+r/hQB6zk01xlfrXlH/CldV/6KN4i/7+r/hS/wDCltV/6KN4i/7+L/hQB85eIdB1yx+LNx4f+03El614IIpCzLuR2BB4JIXBzX2zpOnLpuk2logCiCFI8A56ACvOPDHwRs/DvjiLxTe65f6vfxghXvMHkrszwOcKSP8A9VeqDoKAFooooAKKKKACiiigAooooAKpa1/yAb//AK9pP/QTV2qWtf8AIBv/APr2k/8AQTQBxvwN/wCSK+H/APrnL/6Okrv64D4G/wDJFfD/AP1zl/8AR0ld/QAUUUUAFFFFABXCfGn/AJI14k/69h/6MWu7rhPjT/yRrxJ/17D/ANGLQB1Xh/8A5FrS/wDr0i/9AFaNZ3h//kWtL/69Iv8A0AVo0AFFFFABRRRQBm+Iv+RX1T/rzm/9ANHhz/kV9L/684f/AEAUeIv+RX1T/rzm/wDQDR4c/wCRX0v/AK84f/QBQBpUUUUAFFFFABRRRQAUwrnnFPooAiCAfSpFHFLiigAooooAKKKKACqeq/8AIIvP+uD/APoJq5VPVf8AkEXn/XB//QTQBw/wJ/5Ir4f/AOucn/o1q9EFed/An/kivh//AK5yf+jWr0QUAFFFFABRiiigAooooAKKKKACvNfhZ/yNfxB/7Dx/9FrXpVea/C3/AJGv4g/9h4/+i1oA9KooooAKKKKADFFFFABRRRQAUUUUAFFFFABRRRQAVS1r/kA3/wD17Sf+gmrtUta/5AN//wBe0n/oJoA434G/8kV8P/8AXOX/ANHSV39cB8Df+SK+H/8ArnL/AOjpK7+gAooooAKKKKACuE+NP/JGvEn/AF7D/wBGLXd1wnxp/wCSNeJP+vYf+jFoA6rw/wD8i1pf/XpF/wCgCtGs7w//AMi1pf8A16Rf+gCtGgAooooAKKKKAM3xF/yK+qf9ec3/AKAaPDn/ACK+l/8AXnD/AOgCjxF/yK+qf9ec3/oBo8Of8ivpf/XnD/6AKANKiiigAooooAKKKKACiiigAooooAKKKKACiiigAqnqv/IIvP8Arg//AKCauVkeItUstO0a6fULy3tEeJ0RriURhm2ngZ70Acl8Cf8Akivh/wD65yf+jWr0QV5j8B9UsZfhPolhDeQPdRRSNJAsimRB5rclQcgcjk+tenUAFFFFABRRRQAUUUUAFFFFABXmvwt/5Gv4g/8AYeP/AKLWvSq81+Fv/I1/EH/sPH/0WtAHpVFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVLWv+QDf/APXtJ/6Cau1S1r/kA3//AF7Sf+gmgDjfgb/yRXw//wBc5f8A0dJXf1wHwN/5Ir4f/wCucv8A6Okrv6ACiiigAooooAK4T40/8ka8Sf8AXsP/AEYtd3XCfGn/AJI14k/69h/6MWgDqvD/APyLWl/9ekX/AKAK0azvD/8AyLWl/wDXpF/6AK0aACiiigAooooAzfEX/Ir6p/15zf8AoBo8Of8AIr6X/wBecP8A6AKPEX/Ir6p/15zf+gGjw5/yK+l/9ecP/oAoA0qbuweadXnfxkutQtfCNimkX8+nz3Wr2tqZ4G2squxU0Aehg5pa8kHwi8Uf9FN13/voUv8AwqLxR/0U7Xf++hQB61RXkv8AwqLxR/0U7Xf++hR/wqLxR/0U7Xf++hQB61RXkv8AwqLxR/0U7Xf++hR/wqLxR/0U7Xf++hQB6zmm78HmvKP+FReKP+ina7/30Kf8ME1fTPiB4u8Pavrt5rKaaLXypbpskb4g54/4Fj8KAPVS4FAbIyOlQXORayYOCEJGPpXi+gfD7xLq/hHTdbn+JOs2q3llFdMgcBY96BiMnsM0Ae4ZozXj8fwv1+a5kt4vinrDTRgFo1lUsoPQkVKfhJ4o/wCim66P+BigD1vNeWfH/wAMSeIvhlcywFvtGmyi7RQeHAyrA/RWJ+oqv/wqPxRn/kpmuY/3hVT/AIVZrd8s0K/E3V7lBmORFdWGe6mgDmv2XfC7RW+peJ52YCVfscC54IyGYkfVV/M19Fg8CvGNN+CetaTZpZ6Z4/1e0t0zsihwFGTn+dXP+FS+KAP+Sm65/wB9CgD1ujNePn4X+INgf/haWsbS+wN5i4LdMfWpv+FSeKD0+Juu/wDfYoA9a3UKcivHdR+F3iix0u6uh8SdcfyIXk2lwN20E4/SvT/DM0lx4T0meZ2kklsoXd2OSzFASTQBqUUUUAFFFFABXmvwt/5Gv4g/9h4/+i1r0qvNfhb/AMjX8Qf+w8f/AEWtAHpVFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVLWv+QDf/8AXtJ/6Cau1S1r/kA3/wD17Sf+gmgDjfgb/wAkV8P/APXOX/0dJXf1wHwN/wCSK+H/APrnL/6Okrv6ACiiigAooooAK4T40/8AJGvEn/XsP/Ri13dcJ8af+SNeJP8Ar2H/AKMWgDqvD/8AyLWl/wDXpF/6AK0azvD/APyLWl/9ekX/AKAK0aACiiigAooooAzfEX/Ir6p/15zf+gGjw5/yK+l/9ecP/oAo8Rf8ivqn/XnN/wCgGjw5/wAivpf/AF5w/wDoAoA0q87+M3/It6H/ANjDYf8Ao2u7vL62sIjLeTxwR5A3SMFGT2ya4D4wyeZ4Z0Mj5gfEFgQR0/1tAHo4FGM1GWPGDjPfFRW15DdA/Z545dp2tsbdg+lAFnFLio91MM4WURs6h2GQueTjr/OgCalxVAatZNcTQrdwmSEEyr5gygHXI7U3+29O+zC4F/bGAvs80Sjbn0z0oA0cV5r4P/5Lv8Q/92w/9J0ruo9YspboW8d5A0xAIjEg3HIz0+lcJ4P/AOS8fEP/AHbD/wBJ0oA9Eux/ok2f+ebfyrz23DS/s42aopdj4ciGF6n9wvSvQrr/AI9Zv+ubfyrmvALRR/Cjw09wyiNdHtSxfoB5K9aAOeitrzR9L8aavaXbm9Z38gug+TbGGXB6kfNj8Kju9Q1XTYXsdT1+a3gS8YNqTKNwHkqyp+LEivSTHHtIZAVbO4EcN9aqXdzp0JMd/JbKcbysu3GOgPPv3oA47QLi7ufEyPfapcLJcaajJat92TlwXHvjBrl9OlvND01VbW5rLzZbudHaMBrmYXDKEPHcAn1r2NI4XCyIqMQMBgo6egpJLe3fG+GNgpyuVBwfWgDhT4quheNYy3Bj1D7QWNsF+ZY/J3flurLj1fW7TR0/tXWpVS7itZZbtlx9m8wEtj2/livTvs8Hm+aYYzL/AH9g3Y+tK1vCybHjjYdCCgx+VAHnmm3E0HgXS2ilNwX1eNTK0fLq033vbI5zWj4RudYfUo5dSv5LmK8jlbyXXAhKyFQB+AFdjF9ndTFGI9sbYKKAdrf0qdYUXG1QuPQUAUPEHPhjU/8Arzl/9ANQ+Ef+RJ0P/sHW/wD6LWpvEH/Is6p/16S/+gGofCP/ACJOh/8AYOt//Ra0AbFFFFABRRRQAV5r8Lf+Rr+IP/YeP/ota9KrzX4W/wDI1/EH/sPH/wBFrQB6VRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAVS1r/kA3//AF7Sf+gmrtUta/5AN/8A9e0n/oJoA434G/8AJFfD/wD1zl/9HSV39cB8Df8Akivh/wD65y/+jpK7+gAooooAKKKKACuE+NP/ACRrxJ/17D/0Ytd3XCfGn/kjXiT/AK9h/wCjFoA6rw//AMi1pf8A16Rf+gCtGs7w/wD8i1pf/XpF/wCgCtGgAooooAKKKKAKGswy3eiX9rAoaWa3kjQE4yxUgc/U15bZaj8bbDT7e0i8MaC0cESxqWu+SFAAz8/tXr5XJOaUDFAHzL8Z9T+JV14BKeMtG0ux077VGTLZz7nD84H3jx1rhvCfi3xfqdlpWmX7XV/oKa5Zn7VMhcQyCQEL5nvnoSenGK+uPFfhHS/GWlpp2txvLbJOk+xWxll6A+3PSuM+KmmWmkeDPD1lp1vHbW8OvWCRxxjAUeZ0oA9Eugq2Uu/fs8s5KDLAY7CvGz4kGieH7tPDt7ZtCt2scmqWUkUQVSu7948mYw+cA5Fe1GPP6VH9njCkFEA6n5R19aAPNoda8Q3ttaXces7Et7GCeRLcQzJdM8m07nVSMY5+Q4pNE1aXU/iPB9o1pbqeL7WkmmgR/wChgOgQ8AMMj+8T7V6YIlKjYF2kDGBS+QofcqKGz1xQB49e2Txahr1zPBZxpcx6mtvPBFh3OeVmP8Xbb0/iqDUEi0vTYrbVP+EfsZV1NWkS6iUWIUw4Vtme/r617JMYo9oldEDttXcQNzHsPepGiWUfMqsD6jNAHmPhzRP7W8dX+qWjaV9itbuJ0ltYP3rD7NGAsbj/AJZ89PrVnwd/yXf4h/7th/6TJXosdusedgxnk4+mK868Hf8AJePiH/u2H/pOlAHol1/x6zf9c2/lXncDFf2cbEowQ/8ACOwHdnofIXr6V6Jdf8ek3/XNv5Vzvw7jB+F3hjPOdItev/XFaAOX07xcNPCzXfiiPUNGS78ttUm8oI4MZOwsgC8MB6Hmua1XS7rxFqsOqXGszSeeNOUp9ngdSsk7AqCyE47+nPOa9iF/pUgnhF3ZsIAWmUSKdg7lh26d6Q6tpAt1na/s/JLBVkMy7dw5ABzjNAHl+m+MdcnvNahTUYle3hl2W32m2drcq21WESjeOP72a63xRqUmi+DJF/4SNINQgEbSXc8kEUh3HOPnXYCRkDI7V0rXunC7SD7VbedOu5YjIu51PcDqRSjTraO8uLooGe4C+YWGeFGAPb1oA8yfxtqb+JNGtrfVAlrNBAyLc3VtF9r3j5jhhuftgpx+Yq5LrviDT9Ou9Qm1h5o5YrpkV4YwloI5AoYYX0PO4npXpXkR7gxRflAwfTFL5abeQuMHkgd+tAHG/DicXVtrMq6r/a8b3x2X3y/vhtXn5AF9uB2ruc8VVjaFJWij8sOuCyjAIz0P6VKJ0LMgYF1A3KDyKAKfiD/kWtU/685f/QDUPhD/AJEnQ/8AsHW//otadrkyTeGNTeJ1dfskw3Kcj7pFR+EZB/whOh/9g+D/ANFrQBtUUmc0tABRRRQAV5r8Lf8Aka/iD/2Hj/6LWvSq81+Fv/I1/EH/ALDx/wDRa0AelUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFUta/5AN//ANe0n/oJq7VLWv8AkA3/AP17Sf8AoJoA434G/wDJFfD/AP1zl/8AR0ld/XAfA3/kivh//rnL/wCjpK7+gAooooAKKKKACuE+NP8AyRrxJ/17D/0Ytd3XCfGn/kjXiT/r2H/oxaAOq8P/APItaX/16Rf+gCtGs7w//wAi1pf/AF6Rf+gCtGgAooooAKKKKACiiigArzv4zf8AIuaH/wBjFYf+ja9Erzv4yjPhzQ/+xhsD/wCRaAPQxSEZ7ZBpQfXilzQAgGFAAopc0UAV57ZLgxmREby3DruXOD6ipwOKM0uaACvNfB//ACXj4h/7th/6TpXpWa808H/8l4+If+7Yf+k6UAeiXX/HpN/1zb+VYHw6H/Fr/C//AGCLX/0Utb91n7JN/wBc2/lWD8Of+SX+F/8AsEWv/opaAOAvrC5hi1+We3tY1ubXURbywoQ0p3ciXJ+Y9MYx3qvdQtpy29vfppFpNHqjNLFPExtADAMFVznn69a9je2hI+aNSvPykDHPX86bLY285/fwRyj/AG0BoA8subzRojdG7lhTWZNSt/sbDIbyyYtvl+iYyOOPvVr6/f3mn+I9Xkt9YmjcWEbw2ZVSgGcFxxklevXvXa3ej2F4hS6s4JVK7fmjHT0qb7Fb4A8iM7U2KSoJC46fSgDh4db1BPBeuT2OrtqTW3FtqDouWyBnhQAcH2rK8ReINe0PShaNqc8zC8MbX5McThdmRklSuM+1enR2sEcJhjhjSMjJVUAB/ClltbedNs0Mcinkh0BGaAPLda8S6vb6fdzpci3uGtrFjPFsHLeZu+dlI7DqDUumNrQ1L+0hrUk6+XaLOqxoUudzMCSQOwOOMV3OteGrTXLJbaVpLXawYSWxCMMdvpzVzTdKttL0+K0gXckQA3Pyzc5yfxNAHjya1qdh4S1WGw1Jr4m0ujLbsikWX7xgMYGeR6k9a2fBepapf6R4dsLbVpGtdqq10qKWkVYQ3l9MYB+Xp2rvtesraPw3q3lQRoXtJtxVAN3yHrTPCFvCvgzRGSJF/wBBhYYUDBMYzQBugYGKdTadQAUUUUAFea/C3/ka/iD/ANh4/wDota9KrzX4W/8AI1/EH/sPH/0WtAHpVFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVLWv8AkA3/AP17Sf8AoJq7VLWv+QDf/wDXtJ/6CaAON+Bv/JFfD/8A1zl/9HSV39cB8Df+SK+H/wDrnL/6Okrv6ACiiigAooooAK4T40/8ka8Sf9ew/wDRi13dcJ8af+SNeJP+vYf+jFoA6rw//wAi1pf/AF6Rf+gCtGs7w/8A8i1pf/XpF/6AK0aACiiigAooooAKKKKACuZ8c+D4fG+gLpdxdy2apcJcLLCPmVkORXTUmKAPJv8AhR05/wCZ717/AL+f/Xo/4UbP/wBD5r//AH8/+vXrQGKKAPJf+FGz/wDQ+a//AN/P/r1m+IfhFd6N4b1HUovG+uPJaW0kyI0vBKqTzz7V7ZWB45H/ABQOvH/qHzf+gGgDyvwf8J7zxH4L0jWbjxvrkUt/aR3DokvCllBwOfetn/hRs/8A0Pmv/wDfz/69dd8LP+SS+F/+wXB/6AK6ygDyX/hRs/8A0Pmv/wDfz/69dF4F+HUPge+1G7XVbvU59QCebLdcsdgwOfpxXcUhGaAIZozLE6E4DqVPtXlEPwIa3gjht/G2uQwxqFSNHwqKBgADPSvXdvrS4oA8k/4UbOT/AMj3r3/fz/69A+Bkw/5nzXv+/n/169bxRQB5KfgbP/0Pev8A/fz/AOvUF58FZ7axmmXxzrrNGjOMy9cDPrXsNU9VA/si84/5YP8A+gmgDwr4e/Da+8Y+BNO1678Z63BNeKzNHHKSBhyvr7V0/wDwo2b/AKHzX/8Av5/9etb4Fc/BXQD/ANM5P/RrV6IKAPJf+FGTf9D3r3/fz/69J/woyfr/AMJ5r3/fz/69et0UAeRv8C5ZI2STxzrrI4IZTJkEfnXqOmWKaZpNpYRsXS1gSFWPUhVCg/pVujFABRRRQAUUUUAFea/C3/ka/iD/ANh4/wDota9KrzX4W/8AI1/EH/sPH/0WtAHpVFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABVLWv8AkA3/AP17Sf8AoJq7VLWv+QDf/wDXtJ/6CaAON+Bv/JFfD/8A1zl/9HSV39cB8Df+SK+H/wDrnL/6Okrv6ACiiigAooooAK4T40/8ka8Sf9ew/wDRi13dcJ8af+SNeJP+vYf+jFoA6rw//wAi1pf/AF6Rf+gCtGs7w/8A8i1pf/XpF/6AK0aACiiigAooooAKKKKACiiigAooooAKwPHTf8UDrw7/ANnzf+gGt+q97ZQ6hZTWt3GJIZ0KSIejKRgigDgPht418LWPwx8OWt74l0e3uIdOhSSKW/iVkYIAQQW4NdN/wn/g7/obND/8GUP/AMVWN/wpj4fnr4Xsf++aX/hTHw+/6Fex/wC+aANj/hP/AAd/0Nmh/wDgyh/+Ko/4T/wd/wBDZof/AIMof/iqx/8AhTHw+/6Fex/75o/4Ux8Pv+hXsf8AvmgDY/4T/wAHf9DZof8A4Mof/iqP+E/8Hf8AQ2aH/wCDKH/4qsf/AIUx8Pv+hXsf++aP+FMfD7/oV7H/AL5oA2P+E/8AB3/Q2aH/AODKH/4qj/hP/B3/AENmh/8Agyh/+KrH/wCFMfD7/oV7H/vmj/hTHw+/6Fex/wC+aANj/hP/AAd/0Nmh/wDgyh/+KqrqXj3whJpl0ieKtFZmhcALqERJO09Pmqj/AMKY+H3/AEK9j/3zTf8AhTHgDP8AyK9l/wB8n/GgDnvgz4u8N6V8I9EstS8Q6TaXMUcgeGa+iRlPmMeQW9DXdjx/4O/6GzQ//BlD/wDFVjD4MeAD18MWX4qaX/hTHw+/6Fex/wC+KANj/hP/AAd/0Nmh/wDgyh/+Ko/4T/wd/wBDZof/AIMof/iqx/8AhTHw+/6Fex/75o/4Ux8Pv+hXsf8AvmgDY/4T/wAHf9DZof8A4Mof/iqP+E/8Hf8AQ2aH/wCDKH/4qsf/AIUx8Pv+hXsf++aP+FMfD7/oV7H/AL5oA2P+E/8AB3/Q2aH/AODKH/4qj/hP/B3/AENmh/8Agyh/+KrH/wCFMfD7/oV7H/vmj/hTHw+/6Fex/wC+aANj/hP/AAd/0Nmh/wDgyh/+Ko/4T/wd/wBDZof/AIMof/iqx/8AhTHw+/6Fex/75o/4Ux8Pv+hXsf8AvmgDY/4WB4O/6GzQ/wDwZQ//ABVeffDbxf4bs/E3jqS88QaXbpc60ZYWlvY0EqbANykn5h7iuo/4Ux8Pv+hXsv8AvmuF+H3wz8H6r4i8Z299oNrPFY6wYLdWXiNNgO0fnQB6n/wn/g7/AKGzQ/8AwZQ//FUf8J/4O/6GzQ//AAZQ/wDxVY3/AApj4f8A/Qr2P/fNL/wpj4ff9CvY/wDfNAGx/wAJ/wCDv+hs0P8A8GUP/wAVR/wn/g7/AKGzQ/8AwZQ//FVj/wDCmPh9/wBCvY/980f8KY+H3/Qr2P8A3zQBsf8ACf8Ag7/obND/APBlD/8AFUf8J/4O/wChs0P/AMGUP/xVY/8Awpj4ff8AQr2P/fNH/CmPh9/0K9j/AN80AbH/AAn/AIO/6GzQ/wDwZQ//ABVH/Cf+Dv8AobND/wDBlD/8VWP/AMKY+H3/AEK9j/3zR/wpj4ff9CvY/wDfNAGx/wAJ/wCDv+hs0P8A8GUP/wAVR/wn/g7/AKGzQ/8AwZQ//FVj/wDCmPh9/wBCvY/980f8KY+H3/Qr2P8A3zQBsf8ACf8Ag7/obND/APBlD/8AFUf8J/4O/wChs0P/AMGUP/xVY/8Awpj4ff8AQr2P/fNH/CmPh9/0K9j/AN80AbH/AAn/AIO/6GzQ/wDwZQ//ABVH/Cf+Dv8AobND/wDBlD/8VWP/AMKY+H3/AEK9j/3zR/wpj4ff9CvY/wDfNAGx/wAJ/wCDv+hs0P8A8GUP/wAVVPV/HnhCTRr2NPFWiMz28iqq6jESx2ngDdyap/8ACmPh9/0K9j/3xR/wpj4f/wDQsWP/AHxQBz3wb8ZeGdM+EmhWWpeIdKtLmNJd8U97HG6/vnPKswPQiu5/4T/wd/0Nmh/+DKH/AOKrF/4Ux4A3f8izZY/3T/jTv+FMfD7/AKFex/75oA2P+E/8Hf8AQ2aH/wCDKH/4qj/hP/B3/Q2aH/4Mof8A4qsf/hTHw+/6Fex/75o/4Ux8Pv8AoV7H/vmgDY/4T/wd/wBDZof/AIMof/iqP+E/8Hf9DZof/gyh/wDiqx/+FMfD7/oV7H/vmj/hTHw+/wChXsf++aANj/hYHg7/AKGzQ/8AwZQ//FVxfxc8ZeGNS+FHiCz0/wARaTdXMtuBHDDexu7nevRQcmtv/hTHw+/6Fex/75pP+FMeAQRjwxZf98mgDqfD5/4pvS/+vSL/ANAFaVQ29sltDHDCoSONQqKP4QBgCpqACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigBpPpXm3wsP/ABVfxB/7Dx/9FrXV+MvED+FvCOoa5HatdGyj8zyV6sNwH9TXhfwh+LQvfiPqOnR6Y+fEmpNdF92fIATv/wB8/rQB9KCikXpS0AFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABSE00yYJyMD1rm7/xjEL6XT9Es5tVvozsdYB+7ifGQsj87MjmgDomchhmmvcKrYLKMepArlBpvi3WWJ1LVItItZPm8mwjDTxHsvmtuUj1O38qm/4V5pVwPM1e41DVLrvcz3bxsR2GIii8fSgDSPjDw8jMr6zZqynBBmHBqld/EDQLWRVW6e6DDO60iaYD2JUHB+taCeFtBRAv9jWBCgAFrZCfzI5q3bafZ6epSwtYLVGOSIYwgJ9cCgDlb/x34a1LT57K6TUHhuI2jdfsMvQjB/h968Z+D2jWHgvxtrOq6vFeCNS0FgyWcp3Rk53fd9MD1r6YEnH4etNABPH1xQBzVv8AEXQJ50iMl1Bu48ye1eNB9WIwK0R4x8OY/wCQ1Zf9/hWncWkN3A0N1Ek0bjDJIu5W+oPWqQ8M6F/0BdO/8BE/woAtw39tdQLNbzxyRuMq6sCCPWpBJkjnn0rnLj4d+G7iZpXsHVmOSI7uaNR9FVwB+AFQN4b8Sacc6J4me5DcNHq8CzKoHTZ5ewj0OSegoA64GlzXIr4xudJIXxbpUunJ1N7H+8tlXoCzjhCTxtPqPWuoiuYp41eFg6MMh15BH1oAmoozRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUHoaKD0OKAOV1WN/EHiJtF86WKwtoRNdtBIUZ2Y/JGT1wRknGQQMHrW5Zada6ZYxWWnW8VrbQjEcUMaoqd+ABgfhWFqzT6D4ibW47eWe0uIBDdpAMurA/LJ7gAkEcYznmtvTdSs9Us1udPuEngflXQ9voeRQBdVSFp4pNw4pc0AN71ynjzVbrSNJtpbS+ewMt0kTzx2puWVTnogBJ/CurrI1/Qxr1pHCL24sXhlWVJ7fG9SPTII/SgDFtPGYivo9OurW+ukQItxqbQrFGGZdwBQkMCQRwF4p+h+KrvWvFklobGe0svsnnQtOUPn/ORvXaSQPY4PtV628KwRRv9puZ7uSSRJJJZSNzlV2jOAB0qPQ/CEOh6k92l/d3R8ryYknZSsSbidq4A7+tAHS0opBS0AFJilzSbhQBGYQQ24Kd3tXIXFjD4L1K2vdMQw6VdXCwXdqrYhtt+QkiJ2Jl2ghRyZCT0zXYSzRwxtJK6oijJZjgCuQuLxPGWpWtpp6yNpdtcLcXVy6/u7nYDtjTvxJtYnjBjxznNAHZ9qKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigBjLuXB5B6g1y2q+DRNqKX+hX8+j3W4vKbc/JM2MAunRsV1lJigDjRfeLtJ+W7sodXt4ztElt8k8noxX7o96ePiBZW/wC71ixv9PuuphaBpOPXKgiutZQSeKUpmgDCg8YeHri3SX+2LSIMM7JpRG4+qnBFX7LVrLUld9Pu4blEIDNBIHAPvimT+G9GuZ3mudLs5pXOWd4FJJ9zis2+8A+H76RWawWAKMYtmMQP1C4zQB0eaBXJ/wDCs/Dn/PG6/wDAuT/Gl/4Vn4c7Q3X/AIFyf40AdBf6ja6bCst9cRW0ZbbvlcKufTJ71U/4Szw9/wBBzTv/AAKT/GqNn8P/AA/ZT+almZzjG25kaVfrhsjPvV8eFdBx/wAgXT//AAGT/CgDGPxE0R8parezzMcRxrauN7dhkjHJpBrvinUuNM8PLYlOXbUJOGz/AHdv+eldeEATaowOwpFTGcjHtQBykfg+bUZFl8UanPqZBwLdT5UG3qA0a8MQecnnp6V1EMCW8SxQIsaKMBVGAPwqTaAaXNAADS0UUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQA09adRRQAUhoooAKWiigAooooAKKKKAENFFFAC0UUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFAH//Z)

**Gambar 2.2** Flowchart Proses Control Message

### **2.2.3 *Route Maintenance***

*Route Maintenance* adalah sebuah proses untuk mengecek segala rute yang ada. Jika ada koneksi yang putus di antara dua node, maka RERR dikirimkan ke beberapa tujuan.



**Gambar 2.3** Ilustrasi terjadinya RERR

Pada Gambar 2.3, terlihat ada 9 node yang beroperasi dengan rutenya masing-masing, lalu ketika node 1 mengirimkan kepada node 4 pengiriman berhasil, namun ketika node 4 mengirimkan ke pada node 6, pengiriman gagal, karena pengiriman gagal akhirnya node 4 mengirimkan RERR kembali ke pada node 1. Lalu ketika *source* *node* mendapatkan RERR, *source* *node* akan memulai untuk melakukan proses *route* *discovery* kembali.

## **2.3 *K-Means Clustering***

K-Means Clustering merupakan teknik pengelompokkan terhadap objek n kedalam kluster k berdasarkan jarak terdekat ke centroid cluster atau pusat dari suatu kluster. Langkah-langkah untuk melakukan K-Means Clustering adalah sebagai berikut

1. Tentukan jumlah cluster

2. Alokasikan data ke dalam cluster secara random

3. Hitung centroid/rata-rata dari data yang ada di masing-masing cluster

4. Alokasikan masing-masing data ke centroid/rata-rata terdekat

5. Kembali ke Step 3, apabila masih ada data yang berpindah cluster atau apabila perubahan nilai centroid, ada yang di atas nilai threshold yang ditentukan atau apabila perubahan nilai pada objective function yang digunakan di atas nilai threshold yang ditentukan.

A close up of a map

Description automatically generated

**Gambar 2.4** Flowchart dari K-Means Clustering

## **2.4 *Backup Routing***

*Backup Routing* merupakan sebuah algoritma yang dikemukakan oleh Struthy dkk. [3] Algoritma tersebut bertujuan untuk memperbaiki atau melakukan *Refinement* terhadap beberapa *Control Message* yang ada pada AODV, seperti *Refined Route Request* (RRREQ), *Refined Route Reply* (RRREP), *Refined Route Error* (RRERR).

### **2.4.1 *Refined Route Request* (RRREQ)**

RRREQ merupakan sebuah modifikasi dan inovasi dari RREQ standar yang ada pada AODV [3]. Ilustrasi dari perubahan RREQ ada pada Gambar 2.5 di bawah ini.



**Gambar 2.5** Ilustrasi proses berjalannya RRREQ

Di RREQ, setiap node memiliki rute masing-masing, terutama memilik rute arah balik ke pada *Source Node.* Akan tetapi, rute arah balik yang disimpan oleh node pada RREQ biasa hanya ada satu rute. Namun di RRREQ, seperti pada gambar, semua node tidak hanya menyimpan satu rute arah balik, melainkan semua rute arah balik yang mungkin dilakukan akan disimpan oleh node. Misalnya, ketika node p mengirim ke node r, *node* tersebut akan menghitung seluruh rute terlebih dahulu, lalu ketika saat mengirim ternyata node tujuan berbeda dengan yang seharusnya, maka paket tersebut akan dikirimkan melalui node lain sesuai rute yang ada. Hal ini diklaim lebih efektif oleh penulis paper [3].

### **2.4.2 *Refined Route Reply* (RRREP)**

Sama halnya dengan RRREQ, RRREP menyimpan seluruh rute yang tersedia dari *Destination Node* menuju *source node*. Berbeda dengan RREP biasa yang hanya menyimpan satu rute ketika *destination node* akan melakukan reply kepada *source node*. Pada Gambar 2.6 akan menggambarkan ilustrasi bagaimana proses RRREP berjalan.



**Gambar 2.6** Ilustrasi proses berjalannya RRREP

Gambar tersebut menjelaskan hamper sama seperti RREP, namun dapat dilihat bahwa pada RRREP setiap node menyimpan seluruh rute yang tersedia sebelum ia mengirimkan kembali paket data. Sebagai contoh, misal node p ingin melakukan *reply* terhadap node d, makai a bisa melewati node s maupun node r karena kedua rute tersebut sudah disimpan dan dapat dilewati ketika dibutuhkan.

### **2.4.3 *Refined Route Error* (RRERR)**

Dalam algoritma ini, seluruh rute pengiriman *forwarding* dan *reversing* disimpan sebelum pengiriman dijalankan. Namun, jika pengiriman gagal, sama halnya pada AODV biasa node akan mencari RERR atau *Route Error*. Namun pada metode *Backup Routing*, RERR dimodifikasi lebih lanjut dan dinamakan RRERR. Ilustrasi proses terjadinya RRERR terdapat pada Gambar 2.7.



**Gambar 2.7** Ilustrasi proses berjalannya RRERR

Dapat dilihat pada gambar, ketika node v mengirimkan kepada node x, terjadi error. Karena terjadi *error*, maka *source node* sebelum mengirimkan paket ke pada node d akan menghitung seluruh rute yang ada dengan asumsi node tujuan adalah d, maka dari itu *source node* akan menggunakan rute selain rute yang sama karena rute yang sama telah terjadi *error*. Berdasarkan gambar, rute tersebut bisa menjadi bermacam-macam, namun dilihat dari tabel pada gambar, ada beberapa factor yang diperhitungkan, yaitu sebuah *value* bernilai a. Nilai a merupakan hasil perhitungan dari suatu rute, dengan menghitung jumlah energi dibagi dengan *hop count*, setelah itu nilai a terbesar dari setiap rute akan dipilih menjadi rute yang dipakai, dengan catatan selain rute yang terjadi error.

**2.5 *Network Simulator-2* (NS2)**

Network Simulator-2(NS-2) adalah suatu aplikasi untuk melakukan simulasi jaringan dengan melibatkan Local Area Network(LAN), Wide Area Network(WAN), dan beberapa perkembangan terbaru telah menambakan jaringan nirkabel dan juga jaringan adhoc. NS-2 menggunakan dua bahasa pemrograman yaitu C++ dan Object-oriented *open*(OTCL). Pada NS-2, bahasa pemrogaman C++ dapat digunakan dalam hal pengaturan mekanisme internal(backend) dari objek simulasi yaitu pengaturan protokol yang digunakan saat melakukan simulasi, dan bahasa pemrograman OTCL mendefinisikan lingkungan simulasi eksternal(frontend) untuk perakitan dan konfigurasi objek. Proses simulasi pada NS-2 akan memberikan output berupa *file* NAM dan trace *file*.

**2.6 OpenStreetMap(OSM)**

OSM merupakan sebuah web untuk membuat sebuah peta dan dapat diakses secara gratis. OSM dibangun oleh sukarelawan dengan melakukan survei menggunakan GPS dan mendigitalisasi citra satelit serta juga mengumpulkan data geografis yang tersedia di publik. Open Data Commons Commons Open Database License 1.0 adalah sarana dimana kontributor OSM dapat memiliki, memodifikasi, dan membagikan data geografis secara luas [4].

Di dalam OSM sendiri ada beberapa atribut yang digunakan dalam membuat peta, seperti jalan raya, stasiun, bandara, kantor, rumah sakit, dan jenis lain yang tersebar di seluruh dunia. Data yang tersedia pada OSM pun dapat digunakan secara bebas dan dapat digunakan untuk kepentingan berbeda.

**2.7 Java OpenStreetMap (JOSM)**

Java OpenStreetMap(JOSM) adalah aplikasi yang dapat digunakan untuk melakukan penyuntingan terhadap data yang didapatkan dari OpenStreetMap [6]. Penyuntingan dapat berupa merapikan data yang ada pada OSM, *filtering,* dan beberapa filter lainnya.

**2.8 Simulation of Urban Mobility (SUMO)**

SUMO merupakan aplikasi atau program *open source* yang digunakan untuk melakukan simulasi lalu lintas. SUMO sendiri saat ini sudah semakin berkembang dan semakin banyak fitur yang dapat digunakan dalam pemodelan dan membaca fornat-fornat yang berbeda. SUMO sendiri dapat mendefinisikan beberapa atribut tertentu, seperti panjang kendaraan, kecepatan maksimum, percepatan maupun perlambatannya, dan fitur lainnya.

SUMO memberikan beberapa *tools* untuk membuat simulasi lalu-lintas pada tahap-tahap yang berbeda. Seluruh penjelasan mengenai tools tersebut terdapat pada Tabel 2.4 dibawah.

|  |  |
| --- | --- |
| Tool | Keterangan |
| netgenerate | Membuat peta seperti *grid*, spider, dan bahkan random network. Sebelum melakukan proses ini, pengguna dapat menentukan kecepatan maksimum dan membuat traffic light pada peta. Hasil dari proses ini akan menghasilkan *file* dengan ekstension .net.xml. |
| netconvert | Program CLI yang berfungsi untuk melakukan konversi dari peta seperti OpenStreetMap menjadi format native SUMO. |
| randomTrips.py | Membuat rute acak yang akan dilalui oleh kendaraan dalam simulasi |
| dua*route*r | Melakukan perhitungan rute berdasarkan definisi yang diberikan dan memperbaiki kerusakan rute. |
| sumo | Program yang melakukan simulasi lalu lintas berdasarkan data yang didapatkan dari netgenereate atau netconvert dari randomTrips.py. Hasil simulasi merupakan *file* hasil *export* untuk dikonversi menjadi format lain. |
| sumo-gui | GUI untuk melihat simulasi yang dilakukan oleh SUMO dengan grafis |
| traceExporter.py | Melakukan konversi output dari sumo menjadi format yang dapat digunakan pada simulator lain. |

**Tabel 2.4** *List Tools* yang tersedia pada SUMO

**2.9 AWK**

AWK adalah bahasa pemrograman untuk melakukan *text processing* dan ekstraksi data. AWK bisa digunakan untuk melakukan filtering terhadap suatu teks seperti halnya perintah grep pada terminal sistem operasi linux. Disamping itu, AWK juga dapat melakukan proses aritmatika seperti yang dilakukan oleh perintah expr.

Dalam penggunaanya AWK pattern dapat diketikkan dalam terminal bersamaan dengan text sebagai paramater, selain itu AWK *script* dapat dibuatkan kedalam suatu *file* dengan beberapa pattern tertentu. Berikut ini adalah struktur dari *file* AWK ditunjukkan pada Gambar 2.7

![A screenshot of a cell phone

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RDiRXhpZgAATU0AKgAAAAgABAE7AAIAAAAIAAAISodpAAQAAAABAAAIUpydAAEAAAAQAAAQyuocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEFsY3JlZG8AAAWQAwACAAAAFAAAEKCQBAACAAAAFAAAELSSkQACAAAAAzgwAACSkgACAAAAAzgwAADqHAAHAAAIDAAACJQAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDE5OjExOjIzIDIwOjEwOjM2ADIwMTk6MTE6MjMgMjA6MTA6MzYAAABBAGwAYwByAGUAZABvAAAA/+ELGmh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMTktMTEtMjNUMjA6MTA6MzYuNzk2PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPkFsY3JlZG88L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgA5wCxAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A7+88V/EXV/iV4m8PeC4vC62mhfZdz6stwJH86Hf1jJBwQ3YcY61Z/wCL3/8AVP8A/wAnaPBP/Jdvif8A9wn/ANJmr0mgDzb/AIvf/wBU/wD/ACdo/wCL3/8AVP8A/wAna9JooA82/wCL3/8AVP8A/wAnaP8Ai9//AFT/AP8AJ2vSaKAPNv8Ai9//AFT/AP8AJ2j/AIvf/wBU/wD/ACdr0migDzb/AIvf/wBU/wD/ACdo/wCL3/8AVP8A/wAna9JooA82/wCL3/8AVP8A/wAnaP8Ai9//AFT/AP8AJ2vSaKAPNv8Ai9//AFT/AP8AJ2j/AIvf/wBU/wD/ACdr0migDzb/AIvf/wBU/wD/ACdo/wCL3/8AVP8A/wAna9JooA82/wCL3/8AVP8A/wAnaP8Ai9//AFT/AP8AJ2vSaKAPNv8Ai9//AFT/AP8AJ2j/AIvf/wBU/wD/ACdr0migDzb/AIvf/wBU/wD/ACdqtZ+K/iLpHxK8M+HvGkXhdrTXftW19JW4MieTDv6yEAZJXseM9K9Srzbxt/yXb4Yf9xb/ANJloA9JooooA828E/8AJdvif/3Cf/SZq9JrzbwT/wAl2+J//cJ/9Jmr0mgAqhq2u6ToNuk+uapZabDI2xJLy4SFWbGcAsQCcA8VfrifjIob4N+JgwBH2JjyPcUAdBpXivw9rs7QaJr2majKoy0dpeRysB64UmqcvxC8FwTPFN4v0GOSNiro+pwgqRwQRu4NcJ43m1dPGPg7U/F2k2OladZ6muNR0y6a7kErgqkTs0cRjiYn5iAwOB0rQtJ/EFr8UPHR8M6Xpt6xSyZ/tl89uQ/kHbgLE4bPuV/qAD0GDV9NudJGqW2oWs2nspcXcc6tEVBwTvBxjI65qCw8TaDqsfmaXrenXqbzHut7uOQbgpYrlSedqscegJ7V53qYSbx/4B8N6tEsWnXFtc6lLaOm1Lm8AD4Zenyszvt6AkccCtvxZdS2XxH8Jrb+HbC7uLiWaG0v3vzDJDiF2lVlELZTYMgBuWA6dSAbS/ELwW8byL4u0FkTG9hqcOFz0yd3Fa97qun6bprajqN/bWligUtdTzKkShiACXJxySAPXIrgNBt4J9G+J0U0Mckcms3YdHUEMPskJ5Hes/S7y4+GscfhzW53n8ManARo9/Mc/Y5CmfskrH+H/nmT/u844APSY/EOjTaHJrMGq2U2lxqzNexXCvCAv3jvBxxjFWrK8h1Cyiu7bzPKmXcnmRNGxHurAEfiK4PxXbjUPhhouhW7i3udQFitvK42xRmOSJ/mY8DhcBerHgA9tLxx4l1fQ7mCLTJdNtY3tpZhNexPcvNIu3bDHBE6yHIJJcBgoHTmgDsaK890zxzrXiq20yDw5Fp1lqFzocWsTfbleVP3hZVjRVZT95Gy5Pyjb8rZ4oSX/iy8+L3hiK6ls9OL6HLcz6a8JmET74hKnmLIMseivghR/C2aAPRdM1ay1m3ln02bzo4p5Ld22MuJI2KOOQOjAjPT0q5Xm2k/EbV9R0G0EtnYw6xqGuz6RCFLvBD5W8tI2SGfAjbgbckjpzVTxL8R/E3hbT9etrmx0u71LR2s5RcL5kUM9vcSGJWEeWYOrjBUvjHIPYgHp9zcR2lpNczbzHCjSP5cbSNgDJwqglj7AEntUWnalZ6vp8V9ptwlzbSjKSIeDg4I9iCCCDyCMGuf0jXtXPj/AFDw5rQspBHp8OoW8tpG6bFeSRDG+5m3EFOGG3P90VX+H0LaY+v6TdMZLtNXubyR0+aMrPK0iAMOA20jKHBHXGCCQDsqKKKACvNvG3/Jdvhh/wBxb/0mWvSa828bf8l2+GH/AHFv/SZaAPSaKKKAPNvBP/Jdvif/ANwn/wBJmr0mvNvBP/Jdvif/ANwn/wBJmr0mgArC8YeFYfGXh2bRbzUb6xtLj5Z/sRjDSr/dJdGwM4PGDx1xkVu1heMPFUPg3w7NrV5p19fWlv8ANP8AYhGWiX+8Q7rkZwOMnnpjJoAzpfh9DqF9Y3HiLX9Y1yOwmW4t7a8aBIRKv3XZYYk3kdt2abD4Amtdc1LVrPxhr9vdamyNclUsmU7AVQANbnAAOOOvfJq3pvja1vPEi6BqOmaho2qSwmeCC/WMidB94o8TuhI7jOR6V0tAHK3Hw90q+0OGw1K5v7y4huTeR6pJPtu0nJ/1iyIFAIGFAAAwoGOKktvBSLq9jqep67q2r3OnB/sRvGhUQM6FGYCKJAx2sRl93WtWDX9LuYNMmgvEePVW2WTAH98fLaTjj+4jHJ9PcVo0AcZZfDr7BZ61bw+K9dI1qRprp2FoW8xgqsy/uMKSqhcYwB0APNa9x4TsdS8Gt4b16WbWLSSIxSS3gTzHGflOUVVDLxggA8A8nmtyigDkPHfh+8vfhlPo+irJNNCLbYgYB5Eiljdgp4G4qhA6ckdOtWJ/Ddn4i1a08Sx3WqafO1k1pJCFEJlhZtxSRHQspDDqpU++K2da1e00DQr3V9RZltbGB55Sq5O1Rk4Hc0aPeXd/pMN1qFktjNMN/kLN5uxT0ywAGcYyBkA9z1oA4y4+Dehy2OkR22ra7YXek232SDUbG9ENy0HaJmVcFR7Adz1JJ3H8C6aNU0rULW6vrW40uCS3R4pgTOjsrMJGYFmJZQ2cgkk5yCRXS1XuL+1tbq0triUJNeO0cCYJLsqFz9MKpOT/AFFAHIWfws0q10GTTJNT1S5/4mJ1OC7lkiWa1uSSxkjZI1AySeCCOSMY4p2sfDDT9c0W+sNQ1jVnm1GWGS9vw0AnnWE7o4z+62Kit82FVee/Jz1Njc3txcXqXun/AGSOGfZbSecr/aY9qnzMD7nzFlweflz3q5QBy6+Fk0rxRceLZ9Z1e8uFsBbS2/lQuskUe5gAkcIctuZmwpyS2AMYWmfDvTL7TPD15/aMUkBvNUvLyGGb/WRxSzM6B/RsHJHbOK6mQuImMSq0gU7VZtoJ7AnBwPfBrI8NeIk8Q2t5utzaXmn3T2d5blw4jlUA8MPvKVZSDgcHkA5FAGzRRRQAV5t42/5Lt8MP+4t/6TLXpNebeNv+S7fDD/uLf+ky0Aek0UUUAebeCf8Aku3xP/7hP/pM1ek15t4J/wCS7fE//uE/+kzV6TQAVxPxkYL8G/ExYgD7Ew5PuK7aqGraFpOvW6Qa5pdlqUMbb0jvLdJlVsYyAwIBwTzQByGp+Gbu11W58a+J/EkTyaRplytm1ppxijs1ZCXmKmR2kbC9MgccCuP06a9m1u60e6vdUfT9T8JS3Zju9WeaaeUPGFl2hj5BYSH5EcqRxgY59ktdMsLHTV06ysba2sVUotrDCqRBT1AQDGDk8e9VrLw1oWmNA2naLp1o1srrAYLREMQf7wXA4BwM464oA8N0HRNK/wCLRTyXd8qSW1w13J/a1wogIsi4APmfuRxnC7QRkYxkVsWuv3Gs+Kbie/8AiHaaBqdhrMludHkimEskazFY4hEbgRyh02/MsJbLcHIzXrSeGtCjtbe1TRdOW3tpvtEEQtECRS/31GMK3J5HNPk0DR5dYXV5dJsX1JBhb1rZDMo6YD43frQB40RrU3w38b603i/XxfaDqepwWsqXagSRx4CB127f4c5UKQScYzXXaHLd2HxB8OQDVL+5i1Xw/LcXUd1dNKjyoYdrqpOEOHYHaAD3FdbF4M8Lw6bPp8PhvSI7K4ZWmtksIhHKVOVLLtwSD0z0p0Pg/wANW17bXlv4d0mK6tFC286WMavCB0CMFyoGT0oAzfiVlvh9qscqZsJLeRNRlXmSG2KN5kkanhnA6AkevJAVszxbPCZPDOkaff37JdRyvDZ2d01qbuOOJfme5VlaNU3KTtyWyOCAa6rxLocXiXwvqWiTyvDHqFs9u0qDJTcpGQO+PSq1poMd/o9hD4s0vSry7sSDEVi82NGXhXTeuUJAHA6dMnrQB5RofjXztE8H2HjXxFPZafeJqKT36X7RGWaG4McKPcrtYDywx3ZUsVXPcGxNp9hP4++Hk83iS61uCOLVI31dL2SJZViCtH86OFOASrMD8+w7s4wPV5vDOg3GnyWFxomnS2ckpme2e0jaN5CclypGCxPJPWppNE0qWOzjl0yzdLFg9orW6kW7AYBQY+UgdxigDyRvEurabB4miutbvXtU8XW+nzXssg3Wdo6x7tpAAjGTtyAMbs9eab46udW8N2/i7T9A8Q6oLG10OLUY5GvGnltJzKyBBNJufa6jdgnI28ECvVrTwvoFh9r+w6Hptt9uGLrybSNPtHU/PgfN1PXPU0weEfDa6Q2lL4e0oacz+Y1mLKPyS397Zt2598UAcvoz3On/ABVsrBdTvrm2v/Dr3k8V1dNKpmSaJQ6qThMh2yFAB9OKv+DEig8ReKI9JKz6fJqkktxM/wArpdlIxJGBzuUAD5jtweAGHI108JeHrS7iv7Dw7pMd/bRhLadbSON4wF2qocLlRjjjoO1QeEPD0+g2+pzX0sT3urahJf3KwZ8qN2VV2pnkgKi8nGTk4HQAHQ0UUUAFebeNv+S7fDD/ALi3/pMtek15t42/5Lt8MP8AuLf+ky0Aek0UUUAebeCf+S7fE/8A7hP/AKTNXpNebeCf+S7fE/8A7hP/AKTNXpNABXG/FO+1XSPAd3quh6vPplzZtG26KKGRZA0ioQwkRuzEjGOfXpXZVxnxVsNU1jwDeaRoejzarc3rRrsjlhjVAsisSxkdeCFI4zz+dAGbb+JNR8O/FE+HNU1/+2NNbSZNQnnu44Y5rDy26uYlRdhB7qDx1rVu/if4dh0TVdQtZLy5fTbP7bJbCxnjleI52yKroC0Zx98fKBkkgVLNoFva+CNV/wCEV8K6bpmo31nJiwmtYFWWQqQEm8slGB6H5iOetcWfBfiPUL3U5JdPvohqvhWfS3l1C9hdoZ2zgbI2KKp4wIxjHJweKANq1+K8Mt54Ps5LC/E2u27TXDjSrkKu2LJEYK5PzlfmG4BeScEGunm8b+H7bVEsLm+aGSR2jSWS3lWB3XO5BMV8ssMH5Q2eDxXHwaB4jF98PNTbRGV9DtJ7K7tmuot0fmQRoJGIYjaGjOdpY4IOM5AyvDHg7W9A1yOwvfh14d1BLS7M8XifMAnkQSbwdhXeZscBiygNg5IHIB2TfFnwQln9rbX4RbeXJIJjDJsYRuEfa23DEMy/KMk5BAwc1saZ4s0bWNVk03T7tnu0gFxseCSPzIicCRCygOucDKkjmvNovC/ib/hQeu+HW8O3Kavez3PlWxubflZZi4bcJduADg85yOAa6Oytdcb4oaPqk3hy9gsI9Bewmnee2Iild4pOVWUsQPLKkgHkjGRzQBsfEfWL7Qfh3q+paSxS7ghHluqhmTLqpYA8FgCSBzyBU97rmjeDLK0stRvbyV3R2jUpNe3Eiry7kIGcqNwycYGQOOBWb8Upm07wRPrVupku9LljuLaNuY3fzFUB1OQRz1xleoIPNHivTtZvfF2lNZ2E0umfZZori6sZooLmN2ZMK0rEOsRCknyvmyq+mKANS68beH7TTLO/a/M8F9CZ7b7LBJcPLGACXCRqzbQCMnGBnnFY7fEvTZvHGi6JpyXF3bapZNdreQWcssbAsix7WVSNvzNuY/KvGSK4zQfD/j/wbpvhu/0vw5b6pNaaSdKvtKlv44XGJN6ypJyuMkjHJxjjP3emh8P63a+O/DetR6JawwRabcWVxbWUqJFY75UdBzgsAoIJVeWHQA8AGzofifw4mg6hq1l4hl1LT/7QkD3ErtKElYj9zFgZZQWUKq564GTTpviR4TttIvNSvNXW0gsZFiuluoJYZoXYZVWidRICQCQNvODiuF0zwp4rOj3lz/Yj2d7beMJfEFvZ3VzCRdwuWHl7o3YK4Dk/NxuC8nkg8e+E/EninSfE+o2Ph+4hvNW061023097i3En7uZpGmkYSbAMEKAGY8dB2APSLTxHo2v317o1pdzG6igDyp5csDGN8qJI3IXcuQcOhIz3rN+H2o319purQX80s407WLqxt5pzmR4o3wpY9z1GepABOc5qnaxa1P8AF+HV5vDt7babJoi2bXMs9sRFKX80gqspYgfcyAfm6ZX5qm8Czf2nqvifU54/JuU1eewKR/LGyRYCNgfecrjLHJ7AgACgDsqKKKACvNvG3/Jdvhh/3Fv/AEmWvSa828bf8l2+GH/cW/8ASZaAPSaKKKAPNvBP/Jdvif8A9wn/ANJmr0mvNvBP/Jdvif8A9wn/ANJmr0mgAooooAKKKKACiiigAooooAhu7S3v7Oa0vYI7i3nQxyxSKGV1IwQQeopmn2FvpdhFZWYdYIRhBJK0hAznG5iT9OeBx0qzRQAUUUUAFFFFADJ4VuLeSCTeEkQoxRyjYIxwwIIPuDkVX0zS7LR7BLLTLdLe3QkhF7knJJJ5JJJJJ5JOTVuigAooooAK828bf8l2+GH/AHFv/SZa9Jrzbxt/yXb4Yf8AcW/9JloA9JooooA828E/8l2+J/8A3Cf/AEmavSa828E/8l2+J/8A3Cf/AEmavSaACiiigAooooAKKKKACiiigCje6Z9t1Cwu/tt5b/YpGfyYJdsdxuUrtkXHzAZyBxggGr1FFABRRRQAUUUUAFFFFABRRRQAV5t42/5Lt8MP+4t/6TLXpNebeNv+S7fDD/uLf+ky0Aek0UUUAebeCf8Aku3xP/7hP/pM1ek15t4J/wCS7fE//uE/+kzV6TQAUUUUAFFFFABRRRQAUUUUAQTX1pbXFvb3FzDFNcsVgjkkCtKQMkKDyxABPHap6gmsrS5uLe4uLaGWa2YtBJJGGaIkYJUnlSQSOO1T0AFFFFABRRRQAUUUUAFFFFABXm3jb/ku3ww/7i3/AKTLXpNebeNv+S7fDD/uLf8ApMtAHpNFFFAHm3gn/ku3xP8A+4T/AOkzV6TXm3gn/ku3xP8A+4T/AOkzV6TQAUUUUAFFFFABRRRQAUUUUAFFUb26v4NQsIbPTftVtPIy3Vx56p9lUKSG2nl8sAuB0zmr1ABRRRQAUUUUAFFFFABRRRQAV5t42/5Lt8MP+4t/6TLXpNebeNv+S7fDD/uLf+ky0Aek0UUUAebeCf8Aku3xP/7hP/pM1ek15t4J/wCS7fE//uE/+kzV6TQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAV5t42/5Lt8MP+4t/6TLXpNebeNv+S7fDD/uLf+ky0Aek0UUUAebeCf8Aku3xP/7hP/pM1ek15t4J/wCS7fE//uE/+kzV6TQAUUVg+L/FcXg7RDqt3pl/f2qMqymyERaLcQoJDuuQSQOM+/HNAG9RXN6V41tdQ8RNoF9p2oaPqvkfaY7W/WP99HnBZHjd0bB6jdkeldFJIkUbSSMFRAWZieAB3oAdRVCLW9Omj02SO5BXVRus/lYGUeWZM4xkfKCecenU4q/QAUUUUAUb3TPtuoWF39tvLf7FIz+TBLtjuNyldsi4+YDOQOMEA1erM8R69a+GfDt5rF+HaC1TcUjA3OSQqqM8ZLEDnjmrdhLdzWMUmo20drcsCXhjm80JzwN2BnjHbr69aALFFFV5b+1h1C3sZZgtzco7wx4OXVNu4/huX86ALFFU9Oub25W5OoWH2Ix3EkcI85ZPNiB+WTj7u4c7TyKuUAFFMnaVbeRrZEkmCExo77FZscAtg4Ge+Dj0NZXhrxDH4isLiUQNbXFndSWd1AzBvKmjOGAYcMOQQfQjgHigDYooooAK828bf8l2+GH/AHFv/SZa9Jrzbxt/yXb4Yf8AcW/9JloA9JooooA828E/8l2+J/8A3Cf/AEmavSa828E/8l2+J/8A3Cf/AEmavSaACuI+MbbfhNrPzqhIhAZugPnpiu3rP1bQNH1+GOLXdJsdTjibdGl5bJMEPTIDA4NAHGaz4bu9Nu9V8b+JfEyC6sNIuLezls9OMcdipGTL5ZkdpHyAcFgD0wK4fVGvZLXxVot1c6n9kuPB0l/FbXWryXMzzqGIc4Y+WW6mJWZCB0xkV7jDplhb6WNNt7G2isBGYhapCqxBDwV2AYxz0xiq1p4a0Kw8r7DounW3kxtFH5NoibEY5ZRgcAnqOhoA8gttE0uPxR8LJftl8IjpVy1xKdWuMRsttDIBu8z5B8xO0YBXAIKgAP8ADmuXPiLW7S+uviPbaXqo1Bre50MRTeef3hTyBC9wY+ARiRYcjGT0avXU8OaJHbWdvHo2nrBYSeZaRLaoFt3/ALyDGFPJ5GKG8N6G+qSam+jae1/KpSS7NqhldSMEF8ZIxxjPSgDxcPrL/AzWvFP/AAlmvLqVl9uhR1vPldVuSFJXGAwC4DLtIDEDjGO60Z7nT/irZWC6nfXNtf8Ah17yeK6umlUzJNEodVJwmQ7ZCgA+nFdOvg7wymlPpaeHNJXT5JBI9oLGIRM46MU24J98U6Dwj4btdQgv7Xw/pUN5bIEguI7KNZIlC7QFYLkALxgduKAMX4orDL4HuYNTGzRpWRdSuFP7yCHepLIDgE5xznK9QGPFU/Hcz3fiXTtIt7++Er2VxcGxtL1rBSAUUTyXKsGVUJI2ANktkjC10fjDw6vizwjf6I83kfakAWTbuCsrBlJHcZUZHcZqRNHh1SKxuvE2laZPqdoS0cixiYQtn70bOoZc4B9umTjNAHkmi+N4dU07wnZeOPE1xpdtfaCbj7al6bMT3QcLh5lKkMqbWC5AO75s5Aq7ZWFnN8XPCl3da7dart0Ocf2ot1LAl1JFPGoOFcKVJPKjKvkE7sg16dN4V8PXGmR6bPoOmS2MTmSO1ezjMSMTksEIwDyecVak0jTZp7SabT7WSWx/49ZGgUtb8Y+Q4+XgAcYoA8g0/wASara+GLuG/wBcvZrdvHEukXWpSy/vYbQEqPnUAR5KouVCgeYSMHBqD4g32s+G9J8a2Gg+IdVWxsNOs72Cc3bSy200krRtD5z7nwygPjduHGCAcH1yy8K+HtMs7m003QdMtLa6AFxBBZxxpMB/fUDDdT1pp8I+G20gaU3h7Sjpwk8wWZso/JD/AN7Zt25564oA5/TDPpvxiuNLj1K9ubS60Nb14rq6aVRKJym9AxwmQeQoC9OKn8CpFDqXiWPSis+mtq9xI07HD/amb99HjncqtkBjt6YwQNx118KaFY3Y1PSvDukR6pBD5dtcC1SJ12x+WqeYqllXaAnGcLxjHFReEPD0vh7T777XLHJd6jfzahceSCI0eQ5KrnnAAAyepycDOKAN+iiigArzbxt/yXb4Yf8AcW/9Jlr0mvNvG3/Jdvhh/wBxb/0mWgD0miiigDzbwT/yXb4n/wDcJ/8ASZq9JrzbwT/yXb4n/wDcJ/8ASZq9JoAKKKKAOB+Juq32hXHh27h8UzaDYXmppY3jbLby1jaOV/M3zRttbKAdcY7Z5qKxmvNbtNXbwx8R7vUYLaHZ9sWGxmEU4AcbSkIV1KnDA8jjB64t/EGDWbvVfDJ0fw/eanFpuqpqFxLBPboAixypsAkkUlsyA+mO/apZNS1uSTUJbTwPqFsk1uWmM1zaeddS/KiKgWcqMLklmYcLgAkigCj4K8XX/ivSJvD+uzyaJ4qtrdJJJLZUPnRNgrcwh1ZWVuhBU7TkccVki48UvN4Ai/4TTU1/4SCJzestpZZBFq04KZg+Xkbec8e/J1tR8Jah4i8H6NfWtvJ4f8WaLCv2KWdo3KOqhWjcxsytE+MdehBwDxVK80/xFZXvw+Nt4Yvr+PQLctevBcWy4ZrRodiiSVSSGbJ6DHQmgC7d61r/AIK8aaHYazqh1vRddnazjnnt447i1n25QExhUdWwR90EV3GpTy2ulXc9sqtNFA7xq3QsFJAP41xV1ouu+NPGOiahrWmHRNG0OdruK2mnjluLq4xtQsIyyIq5J4Ykk10/iqyhvvCuoRXJlEawNIfKPJKjcBjow45Ugg9CCKAMDwn4gtdM+Gmja54h1C7luNYihnkMokmkkuJYwTHFEoJxwdqIvQE46mtiLxt4fm0MavFflrVpzbACCTzTMCQYvJ2+ZvGD8m3PB4rmZ7nxD4k+Hfg3VbLTluJ7gW15eraCFZoUaAkvAZiERssBnOQrMBzzXOweEfGuhxnUdE0pLi703xDc6hDaXmoBzfW80XlkeaxJDgHq+Oc9hyAdVrPxW0eyTRH0sXV9HqeoGzdorCdzBsz5isgXeJBjGwjd1OMA1q2HiHw+dd8R3EXiCSV7COE6jbzSnyLAAPjAIAUnDbhkn5RnFc1qWjeJNWt/DusyeGLXT7yz11b+50u0uY3fYYWRneU7Ud9xBOP4cdSMVnat4Q8Ta7r/AI9CaU1imrw6c9hdTzxNDLJaNuKMFcuFc8AlR8ucgHAIB3MfxA8MvHevNqRtPsMIuLhL63ltnWInAkCyqpZScDcARkgd6l0vxx4e1q+s7TTr5pZb62+1Wpa2lRJ48Ako7KFYgMMqDkdwK5HxFpHiHxRO+sN4cuLC4s9FvbGO0a5t2ku5rgIowwk2iNdpOWIPP3ag0fSfElrcfDRJ/DF6qaHp7WuoSfabUiFmiEGeJcsBs3naD8rDGWyoAOl8IXd5H4t8VaJLPcXFjpk1t9le6kMkg82EO67zyyg4wSSeSM8cdfXG6NN/aXxV8Sm4j8t9JitbeBovlEqSR+YS5HLkNkAHIUcgAkmuyoAKKKKACvNvG3/Jdvhh/wBxb/0mWvSa828bf8l2+GH/AHFv/SZaAPSaKKKAPNvBP/Jdvif/ANwn/wBJmr0mvNvBP/Jdvif/ANwn/wBJmr0mgAooooAKKKKACiiigAoIyMHkUUUAUdJ0XT9CtZLbSbf7NbvKZfJV2KITjIRSSEXj7q4GcnHJq9RRQAUUUUAFFFFAFHStFsNFhlj0238rzpDLK7OzvK+ANzOxLMcADJJ4AFXqKKACiiigArzbxt/yXb4Yf9xb/wBJlr0mvNvG3/Jdvhh/3Fv/AEmWgD0miiigDzbU/ht4n/4TvXPEnhTx5/YP9tfZ/tFv/Y8Vz/qYhGvzO3+8eAPvd8Uf8IT8Tv8Aorn/AJbVt/8AFUUUAH/CE/E7/orn/ltW3/xVH/CE/E7/AKK5/wCW1bf/ABVFFAB/whPxO/6K5/5bVt/8VR/whPxO/wCiuf8AltW3/wAVRRQAf8IT8Tv+iuf+W1bf/FUf8IT8Tv8Aorn/AJbVt/8AFUUUAH/CE/E7/orn/ltW3/xVH/CE/E7/AKK5/wCW1bf/ABVFFAB/whPxO/6K5/5bVt/8VR/whPxO/wCiuf8AltW3/wAVRRQAf8IT8Tv+iuf+W1bf/FUf8IT8Tv8Aorn/AJbVt/8AFUUUAH/CE/E7/orn/ltW3/xVH/CE/E7/AKK5/wCW1bf/ABVFFAB/whPxO/6K5/5bVt/8VR/whPxO/wCiuf8AltW3/wAVRRQAf8IT8Tv+iuf+W1bf/FUf8IT8Tv8Aorn/AJbVt/8AFUUUAH/CE/E7/orn/ltW3/xVGmfDbxP/AMJ3ofiTxX48/t7+xftH2e3/ALHitv8AXRGNvmRv908g/d7ZoooA9JooooA//9k=)

**Gambar 2.7** Struktur kodingan pada file AWK

***(Halaman ini sengaja dikosongkan)***

# BAB III PERANCANGAN

Perancangan merupakan bagian yang penting dari implementasi sistem sehingga bab ini secara khusus menjelaskan perancangan sistem yang dibuat pada Tugas Akhir. Bagian yang dijelaskan pada bab ini berawal dari deskripsi umum hingga perancangan implementasi.

## **3.1 Deskripsi Umum**

Pada Tugas Akhir ini akan diimplementasikan sebuah *routing protocol* yaitu AODV, dengan melakukan beberapa modifikasi pada *route discovery* menggunakan metode *K-means Clustering* dan *Backup Routing* lalu akan disimulasikan menggunakan NS-2. Ilustrasi proses simulasi AODV modifikasi ini dapat dilihat pada Gambar 3.1.

A close up of a logo

Description automatically generated

**Gambar 3.1** Diagram Simulasi AODV

Secara umum alur simulasi antara AODV asli dan AODV modifikasi tidak ada perbedaan. Dalam Tugas Akhir ini aka nada beberapa istilah yang akan sering digunakan, berikut ini adalah beberapa istilah yang sering digunakan pada Tugas Akhir ini seperti pada Tabel 3.1.

|  |  |  |
| --- | --- | --- |
| No | Istilah | Penjelasan |
| 1 | AODV | Adhoc On-demand Distance Vector. Protocol yang akan digunakan pada Tugas Akhir ini. |
| 2 | PDR | Packet Delivery Ratio. Rasio jumlah pengiriman paket yang terkirim |
| 3 | E2E | Average End-to-End Delay. Jeda waktu rata-rata yang diukur saat paket terkirim hingga sampai pada tujuan |
| 4 | RO | *Routing* Over*head*. Jumlah *control message* yang terkirim. |
| 5 | RREQ | *Route* *Request*. Paket *request* pada AODV yang dikirim untuk mencari *route* |
| 6 | RREP | *Route* *Reply*. Paket reply pada AODV yang dikirim ke *node* sumber melalui *route* yang sudah dibuat. |
| 7 | RERR | *Route* Error. Paket error pada AODV yang dikirim untuk memberitahukan *node* pada suatu *route* bahwa *route* tersebut tidak bisa digunakan lagi. |
| 8 | BR | *Backup Routing,* merupakan sebuah algoritma untuk aodv, yang akan menghitung seluruh jalur yang tersedia terlebih dahulu sebelum paket dikirimkan. |

**Tabel 3.1** Daftar Istilah

Modifikasi yang dilakukan terhadap *routing protocol* AODV sendiri akan dibagi menjadi 2 bagian, yang pertama melakukan *clustering* pada seluruh node menggunakan metode *K-Means Clustering,* proses *clustering* ini akan dilakukan di awal detik t*,* sehingga tentu data yang diperlukan untuk menggunakan *clustering* harus diperoleh sebelum detik t. Hasil dari pengelompokkan ini akan menghasilkan *n cluster* dan ada *cluster head* dan *cluster gateway* pada setiap clusternya. Lalu modifikasi yang kedua akan menerapkan algoritma *Backup Routing*, dalam algoritma ini maka total energi dari setiap rute yang mungkin pada tiap node akan dihitung lalu akan dibagi jumlah *hop count* pada tiap rute, setelah itu perhitungan tersebut akan menghasilkan nilai a. Nilai a yang mempunyai nilai paling besar akan dipilih sebagi rute untuk mengirimkan paket dari *source node* ke pada *destination node*.

## **3.2 Perancangan Skenario Mobilitas**

Tugas Akhir ini menggunakan 2 jenis skenario, yaitu skenario *grid* dan skenario *real*. Skenario *grid* akan netconvert terlebih dahulu. Tahap berikutnya memiliki tahapan yang sama seperti merancang skenario *grid*, yaitu membuat pergerakan *node* menggunakan randomTrips dan dua*route*r. Kemudian dilakukan penggabungan *file* peta *real* yang sudah dikonversi ke dalam *file* dengan ekstensi .net.xml dan *file* pergerakan *node* yang sudah dibuat sebelumnya. Hasil dari penggabungan tersebut merupakan *file* skenario berekstensi .xml. File yang dihasilkan tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .tcl agar dapat diterapkan pada NS-2.

## **3.3 Perancangan Modifikasi Protokol AODV**

AODV sebagi *routing protocol* yang digunakan untuk MANET dalam Tugas Akhir ini nantinya akan dimodifikasi, menggunakan metode *K-Means Clustering* terlebih dahulu untuk melakukan pengelompokkan dari beberapa node berdasarkan jarak antar node terdekat. Lalu akan ditambahkan metode *Backup Routing* untuk menghitung seluruh rute yang tersedia untuk mencapai node tujuan, lalu dihitung energinya dan akan dibagi dengan hop count lalu menghasilkan nilai a. Nilai a terbesar inilah yang akan dijadikan rute untuk mengirimkan paket dari *source node.*

### **3.3.1 Perancangan K-Means Clustering**

Proses penerapan *K-Means Clustering* dimulai dari mengumpulkan beberapa data yang diperlukan terlebih dahulu. Data yang diperlukan untuk melakukan *clustering* adalah posisi dua dimensi dari seluruh *node* yang berada di daerah simulasi. Data tersebut akan ditampung ke dalam *nodes* *positions* *table* yang ada pada *node*. Variabel yang diperlukan untuk *node position table* terdapat pada tabel 3.2.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Variabel | Tipe | Keterangan |
| 1 | *nodes*\_position\_x | double, array | posisi absis *node* |
| 2 | *nodes*\_position\_y | double, array | posisi ordinat *node* |
| 3 | *nodes*\_*timestamp* | double, array | waktu data diperoleh |

**Tabel 3.2** Nodes Position Table

*Nodes Position Table* sendiri merupakan tabel yang berisi data 2 dimensi posisi dari setiap node yang ada, dan berisi *timestamp* untuk memastikan bahwa data yang akan diambil adalah data yang paling baru sesuai *timestamp*. Oleh karena itu, paket tersebut harus memindahkan data dari node kepada paket tersebut dengan memperhatikan *timestamp.* Proses pemindahan data tersebut ada pada *pseudocode* pada Gambar 3.3.

|  |
| --- |
| for i=0 to *number*\_of\_*nodes* do  if (paket. *positions*\_*table*.*timestamp*[i] > *node*. *positions*\_*table*.*timestamp*[i])  then  *node*. *positions*\_*table*.x[i] = paket. *positions*\_*table*.x[i]  *node*. *positions*\_*table*.y[i] = paket. *positions*\_*table*.y[i]  else if (paket. *positions*\_*table*.*timestamp*[i] < *node*. *positions*\_*table*.*timestamp*[i])  then  paket. *positions*\_*table*.x[i] = *node*. *positions*\_*table*.x[i]  paket. *positions*\_*table*.y[i] = *node*. *positions*\_*table*.y[i]  endif  endfor |

**Gambar 3.3** *Pseudocode* Proses Pemindahan Data Posisi Node

Proses pemindahan data ini akan dilakukan sebelum detik t, yang nantinya akan diteruskan ke proses *clustering*. Proses *Clustering* ini akan dilakukan oleh node yang berperan sebagai *Server Node*. Node tersebut nantinya akan melakukan *clustering* sesuai data dari *Nodes Position Table.* Setelah itu proses ini akan memilih *centroid cluster* awal secara acak sebanyak k *cluster,* lalu setelah itu dapat dilakukan proses *K-Means Clustering. Pseudocode* untuk melakukan *K-Means Clustering* terdapat pada Gambar 3.4 di bawah ini.

|  |
| --- |
| isClusterChange = true  while isClusterChange=true do  for i=0 to *number*\_of\_*cluster* do  isClusterChange = false  for j=0 to *number*\_of\_members\_on\_*cluster*[i] do  for k=0 to *number*\_of\_*cluster* do  distance = euclidean\_distance(*cluster*[i].member[j],*cluster*[k])  if (distance < *cluster*[i].member[j].nearestDistance)  then  *cluster*[i].member[j].setCluster(k)  isClusterChange = true  endif  endfor  endfor  endfor  *update*\_*centroid*\_all\_*clusters*()  endwhile |

**Gambar 3.4** Pseudocode K-Means Clustering

Setelah melakukan *K-Means Clustering* maka akan ada beberapa informasi penting, seperti *cluster head, cluster group, cluster gateway*. Kemudian, *cluster head* dilakukan secara internal dengan pemilihan *node* yang terdekat dari *centroid* *cluster* tersebut sehingga setiap *node* akan dicari jaraknya menuju *centroid* *cluster* tersebut dengan metode euclidean distance. *Pseudocode* untuk pemilihan *cluster head* ini terdapat pada Gambar 3.5. Setelah memilih *cluster head* maka selanjutnya akan dipilih *cluster gateway* dengan mencari titik tengah terlebih dahulu yang ada diantara seluruh node yang ada pada peta simulasi, lalu akan dicari setiap *node* pada tiap cluster yang paling dekat jaraknya dengan *cluster gateway*. *Pseudocode* pemilihan *Cluster Gateway* ini terdapat pada Gambar 3.6.

|  |
| --- |
| *cluster*\_*centroid* = getCentroid\_ThisCluster ()  nearestDistance = 1000000.00  iter = 0  for i=0 to *number*\_of\_members do  distance = euclidean\_distance(member[i],*cluster*\_*centroid*)  if (distance<nearestDistance)  then  nearestDistance = distance  memberId = i  endif  endfor  return member[memberId] |

**Gambar 3.5** Pseudocode Pemilihan Cluster Head

|  |
| --- |
| *centroid* = getCentroid\_All*Node*s()  nearestDistance = 1000000.00  iter = 0  for i=0 to *number*\_of\_members do  distance = euclidean\_distance(member[i],*centroid*)  if (distance<nearestDistance)  then  nearestDistance = distance  memberId = i  endif  endfor  return member[memberId] |

**Gambar 3.6** Pseudocode Pemilihan Cluster Head

Setelah seluruh proses *Clustering* berjalan, maka akan ada beberapa informasi yang dimasukkan kedalam *nodes cluster tables*. Setelah itu, *server node* akan menyampaikan seluruh informasi hasil dari *clustering* sebelumnya ke pada seluruh node yang ada pada peta simulasi. Penerimaan setiap paket RRREP dan *Hello Messages* diikuti dengan melakukan proses pemindahan informasi hasil *clustering* melalui *nodes* *clustering* *table* yang telah disimpan pada paket tersebut. Penjelasan tentang *nodes cluster table* ada pada Tabel 3.3.

|  |  |  |  |
| --- | --- | --- | --- |
| No | Variabel | Tipe | Keterangan |
| 1 | *nodes*\_*cluster*\_group | int, array | *cluster* *node* |
| 2 | *nodes*\_*cluster*\_*head* | int, array | *cluster* *head* |
| 3 | *nodes*\_*cluster*\_*gateway* | int, array | *cluster* *gateway* |
| 4 | *nodes*\_*cluster*\_*timestamp* | double | waktu proses *clustering* |

**Tabel 3.3** Nodes Cluster Table

Dapat dilihat pada tabel, bahwa ada beberapa informasi penting beserta tipenya. Seperti *Cluster Node, Cluster Head, Cluster Gateway,* dan waktu dari proses *clustering.* Proses ini seperti sebelumnya juga memperhatikan *timestamp*, karena hal tersebut akan sangat memengaruhi jika *clustering* dilakukan lebih dari 1 kali, yang nantinya tentu data pada tabel tersebut diganti oleh informasi yang terbaru. *Pseudocode* untuk proses pergantian informasi ini dapat dilihat di Gambar 3.7.

|  |
| --- |
| if (paket.*cluster*\_*table*.*timestamps* > *node*.*cluster*\_*table*.*timestamps*)  then  *node*.*cluster*\_*table* = paket.*cluster*\_*table*  for i=0 to *number*\_of\_*clusters* do  if (*node*.*cluster*\_*table*. *head*[i] == index || *node*.*cluster*\_*table*.*gateway*[i] == index)  then  *node*.isClusterHead = true  break  else  then  *node*.isClusterHead = false  endif  endfor  endif |

**Gambar 3.7** Pseudocode Pergantian Informasi dari Hasil Clustering

### **3.3.2 Perancangan Backup Routing**

*Backup Routing* akan dijalnkan setelah melakukan *clustering,* yakni melakukan pemilihan forwarding node dengan yaitu node yang berhak untuk melakukan rebroadcast paket RREQ. Penentuan forwarding node dilakukan dengan cara membandingkan antara minimum energi dan rata-rata energi yangdimiliki oleh rute RREQ. Apabila nilai minimum energi pada rute lebih kecil nilai minimum energi pada nodeyang dilalui oleh RREQ tersebut, bandingkan rata-rata energi rute tersebut dengan rata-rata energi pada node yang dilalui oleh rute tersebut. Apabila rata-rata energi pada rute lebih kecil rata-rata energi pada node yang dilalui, maka paket RREQ akan diteruskan. Jika rute tersebut tidak memenuhi kriteria tersebut, maka paket RREQ akan di-drop. *Pseudocode* untuk pemilihan forwarding node ada pada Gambar 3.8.

|  |
| --- |
| rq->rq\_min\_energy = energy  rq->rq\_energy = rq->rq\_energy + energy  avg\_energy = rq->rq\_energy / (hop count + 1)  if (rq->rq\_min\_energy < node's minimum energy) then  update node's minimum energy  if (avg\_energy < node avg energy) then  drop RREQ  else  forward RREQ  update node avg energy  end if  else  drop RREQ  end if |

**Gambar 3.8** Pseudocode pemilihan forwarding node pada Backup Routing.

Dapat dilihat pada *Pseudocode* tersebut bahwa energi setiap rute akan dihitung terlebih dahulu, lalu dari setiap node pada rute akan dipilih energi yang terkecil, setelah itu jumlah energi pada suatu rute akan dibagi dengan *hop count* untuk mendapatkan nilai a sesuai pada rumus berikut:

Setelah proses berikut, maka nilai a yang paling besar dibandingkan rute lain akan menjadi rute yang digunakan.

## **3.4 Perancangan Simulasi pada NS-2**

Simulasi dari program yang telah dibuat akan menggunakan NS-2, dengan menggabungkan skrip *tcl* dan skenario yang telah dibuat sebelumnya. *File* gabungan tersebut memiliki konfigurasi lingkungan simulasi yang sesuai pada Tabel 3.4.

|  |  |  |
| --- | --- | --- |
| No | Parameter | Spesifikasi |
| 1 | Network Simulator Tool | NS2 |
| 2 | *Routing* Protocol | AODV |
| 3 | Simulation Time | 200 s |
| 4 | *Number* of *Node*s | 50, 100, 150, 200 |
| 5 | Simulation Area | 700 m x 700 m |
| 6 | *Number* of Cluster | 5,10, 15, 20 |
| 7 | *Number* of Clustering | 1 |
| 8 | Antenna Model | Omni Antenna |
| 9 | MAC Type | MAC/802\_11 |
| 10 | Network Interfaces Types | Wireless |
| 11 | Transmission Range | 400 m |
| 11 | *Source*/*Destination* *Node* | Static |
| 12 | Initial *Node* *Energy* | 100 Joule |

**Tabel 3.4** Spesifikasi Skenario NS2

## **3.5 Perancangan Metrik Analisis**

Dalam Tugas Akhir ini akan ada beberapa Metrik Analisis yang digunakan sebagai data utama untuk membandingkan hasil AODV modifikasi dan hasil AODV asli.

### **3.5.1 Packet Delivery Ratio (PDR)**

PDR merupakan rasio dari jumlah terkirimnya paket dibandingkan dengan jumlah paket yang dikirim. Jika hasil PDR tinggi, maka dapat dikatakan bahwa AODV tersebut mempunyai performa yang baik. Perhitungan PDR ada pada rumus sebagai berikut:

Keterangan:

PDR: Packet Delivery Ratio

*Packet Received*: Jumlah paket yang terkirim atau diterima oleh node tujuan

*Packet Sent*: Jumlah paket yang dikirim oleh *Source Node*

### **3.5.2 Average End-to-End Delay (E2E)**

E2E merupakan rata-rata delay dari waktu pengiriman paket dari *source node* hingga paket tersebut mencapai *destination node*. Jika E2E memiliki waktu yang sedikit, maka dapat dikatakan AODV tersebut memiliki performa yang baik. Rumus dari E2E adalah sebagai berikut:

Keterangan :

CBR(RecvTime) = waktu *destination node* menerima paket

CBR(SentTime) = waktu *source* *node* mengirimkan paket

recvnum = jumlah paket yang berhasil diterima

### **3.5.3 Routing Overhead (RO)**

*Routing* Over*head* adalah jumlah *control message* yang dilakukan per paket menuju *node* tujuan selama simulasi. RO melibatkan semua *control message* meliputi RREP, RREQ, RRER. Jika nilai RO lebih rendah, maka dapat disimpulkan bahwa performa AODV tersebut lebih baik. Rumus dari RO adalah sebagai berikut:

Keterangan :

*packet* *sent* = jumlah paket yang dikirim

sentnum = jumlah paket yang terkirim

### **3.5.4 Average Hop Count (HC)**

HC merupakan hasil rata-rata dari *hop count* di setiap paket menuju *destination node*. Jika nilai HC lebih kecil, maka dapat dikatakan performa AODV tersebut lebih baik. Rumus untuk HC adalah sebagai berikut:

Keterangan :

*hop count* = jumlah *hop count* pada suatu paket

recvnum = jumlah paket yang berhasil diterima

***(Halaman ini sengaja dikosongkan)***

# BAB IV IMPLEMENTASI

Pada bab ini akan diberikan pembahasan mengenai implementasi dari perancangan sistem yang sudah dijelaskan pada bab sebelumnya. Implementasi berupa cara pembuatan skenario mobilitas, modifikasi pada *routing* *protocol* AODV, dan *script* pengujian metrik analisis.

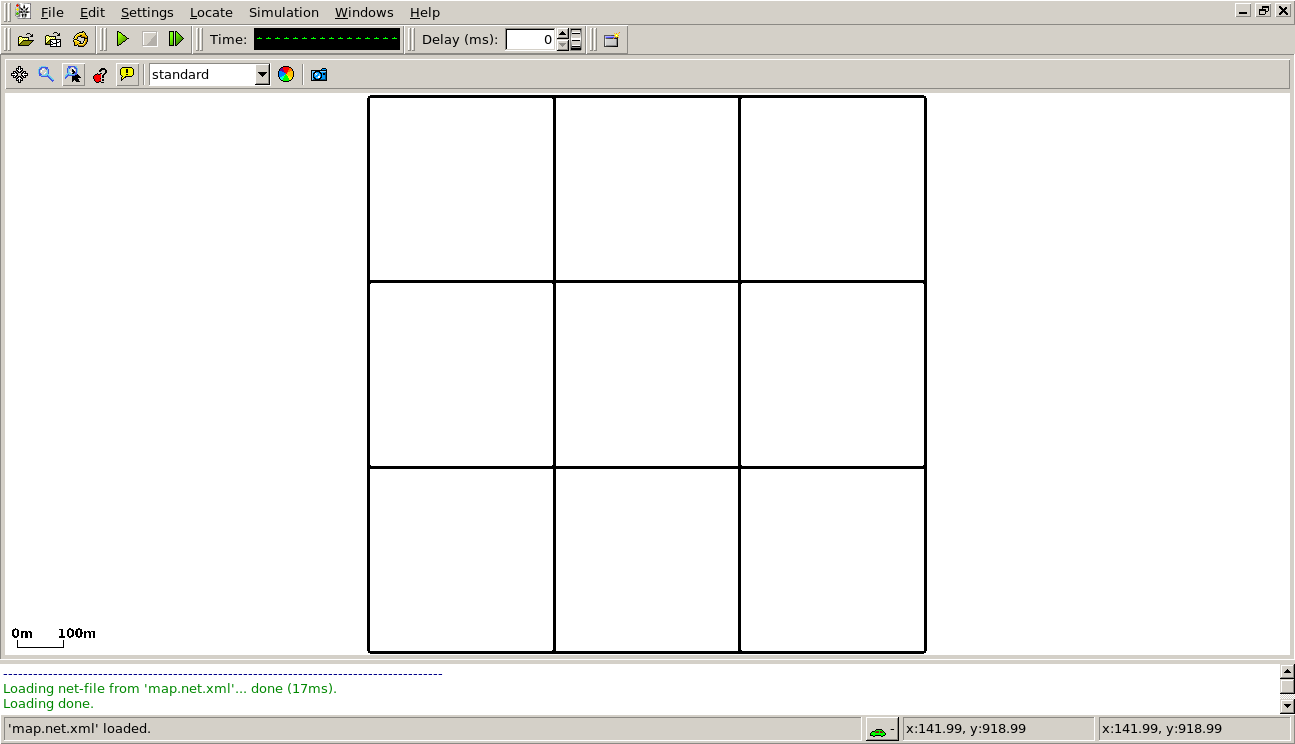
## **4.1 Implementasi Skenario *Grid***

Pada skenario *grid* yang akan digunakan untuk simulasi, SUMO mempunyai tools untuk melakukan generasi peta bernama netgenerate. Dalam Tugas Akhir ini, penulis membuat peta dengan titik horizontal x vertical yaitu 4 x 4 titik. Titik-titik tersebut nantinya akan membuat sebuah 9 petak segiempat. Sehingga untuk membuat sebuah peta seluas 700 m x 700 m, maka dibutuhkan seluas 200 m x 200 m per petak. Perintah netgenerate untuk membuat peta tersebut ada pada Gambar 4.1.

|  |
| --- |
| netgenerate --grid --grid.number=4 --grid.length=200 --default.speed=20 --tls.guess=1 --output-file=map.net.xml |

**Gambar 4.1** Perintah saat melakukan netgenerate

Setelah perintah netgenerate berhasil dilakukan, maka perintah tersebut akan menghasilkan *file* dengan ekstensi .xml. Peta yang telah dibuat ada pada Gambar 4.2.



**Gambar 4.2** Peta Grid Hasil Netgenerate

Setelah berhasil melakukan pembuatan peta, maka dibuatlah *node* serta pergerakan *node* dengan menentukan titik awal dan akhir dari setiap *node* yang telah dibuat. Dalam pembuatan *node* tersebut maka digunakanlah *tools* bernama randomTrips untuk membuat node sebanyak n sesuai pada Gambar 4.3.

|  |
| --- |
| python $SUMO\_HOME/tools/randomTrips.py -n map.net.xml -e 48 -l --trip-attributes="departLane=\"best\" departSpeed=\"max\" departPos=\"random\_free\"" -o trip.trips.xml |

**Gambar 4.3** Perintah dari Tools randomTrips

Setelah melakukan randomTrips, maka dibuatlah rute yang digunakan kendaraan untuk mencapai tujuannya dengan menggunakan *tools* bernama duarouter sesuai pada Gambar 4.4.

|  |
| --- |
| duarouter -n map.net.xml -t trip.trips.xml -o route.rou.xml --ignore-errors --repair |

**Gambar 4.4** Perintah dari duarouter

Saat menggunakan *tools* duarouter, SUMO akan memastikan bahwa kendaraan-kendaraan yang ada tidak melenceng dari jalur peta yang telah dibuat sebelumnya. Maka untuk menjadikan peta dan pergerakan *node* yang telah dibuatmenjadi sebuah skenario dalam bentuk *file* berekstensi .xml, dibutuhkan sebuah *file* skrip dengan ekstensi .sumocfg untuk menggabung file peta dengan rute pergerakan dari *node*. *File* .sumocfg yang akan dibuat sesuai pada Gambar 4.5.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/sumoConfiguration.xsd">  <input>  <net-file value="map.net.xml"/>  <route-files value="routes.rou.xml"/>  </input>  <time>  <begin value="0"/>  <end value="200"/>  </time>  </configuration> |

**Gambar 4.5** Isi dari File Skrip .sumocfg

Setelah *file* .sumocfg dibuat, maka file tersebut akan disimpan di folder yang sama dengan file peta dan file rute pergerakan *node* yang telah dibuat sebelumnya. *File* .sumocfg ini dapat dibuka dengan menggunakan sumo-gui. Setelah itu *file* .sumocfg tersebut dapat dikonversi menjadi file .xml menggunakan *tools* dari SUMO. Perintah untuk melakukan konversi tersebut terdapat pada Gambar 4.6.

|  |
| --- |
| sumo -c file.sumocfg --fcd-output scenario.xml |

**Gambar 4.6** Perintah SUMO untuk melakukan konversi

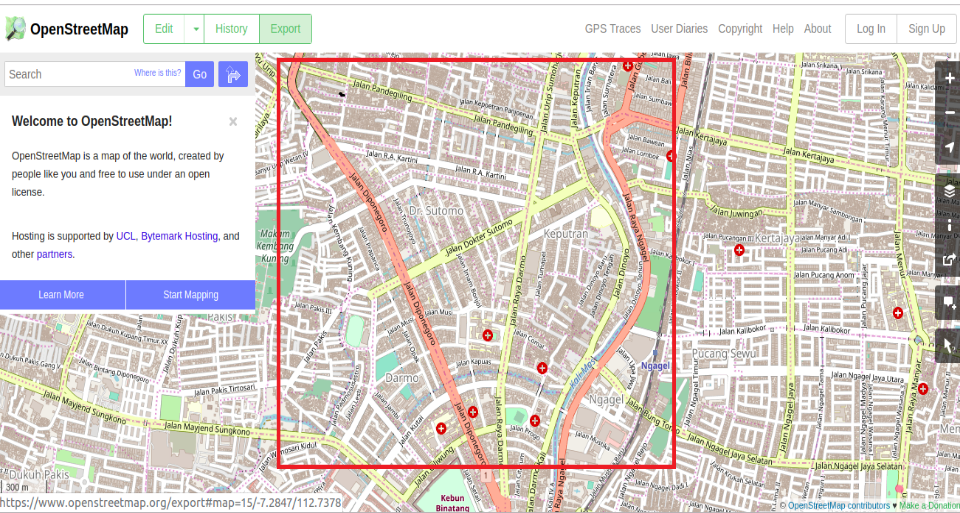
Setelah *file* .xml tersebut telah dihasilkan, maka *file* .xml tersebut akan dikonversi menjadi *file* .tcl agar dapat dibaca dan disimulasikan menggunakan NS-2. Dalam melakukan konversi maka dibutuhkan *tools* bernama traceExporter. Perintah untuk menggunakan traceExporter terdapat pada Gambar 4.7.

|  |
| --- |
| python $SUMO\_HOME/tools/traceExporter.py --fcd-input=scenario.xml --ns2mobility-output=scenario.tcl |

**Gambar 4.7** Perintah dari traceExporter

## **4.2 Implementasi Skenario *Real***

Dalam pembuatan skenario *real*, maka dibutuhkan sebuah area dari dunia nyata yang akan dikonversi menjadi area simulasi. Pada Tugas Akhir ini penulis mengambil area di daerah Jl. Dr Soetomo kota Surabaya. Area yang diambil untuk Tugas Akhir ini terdapat pada Gambar 4.8.



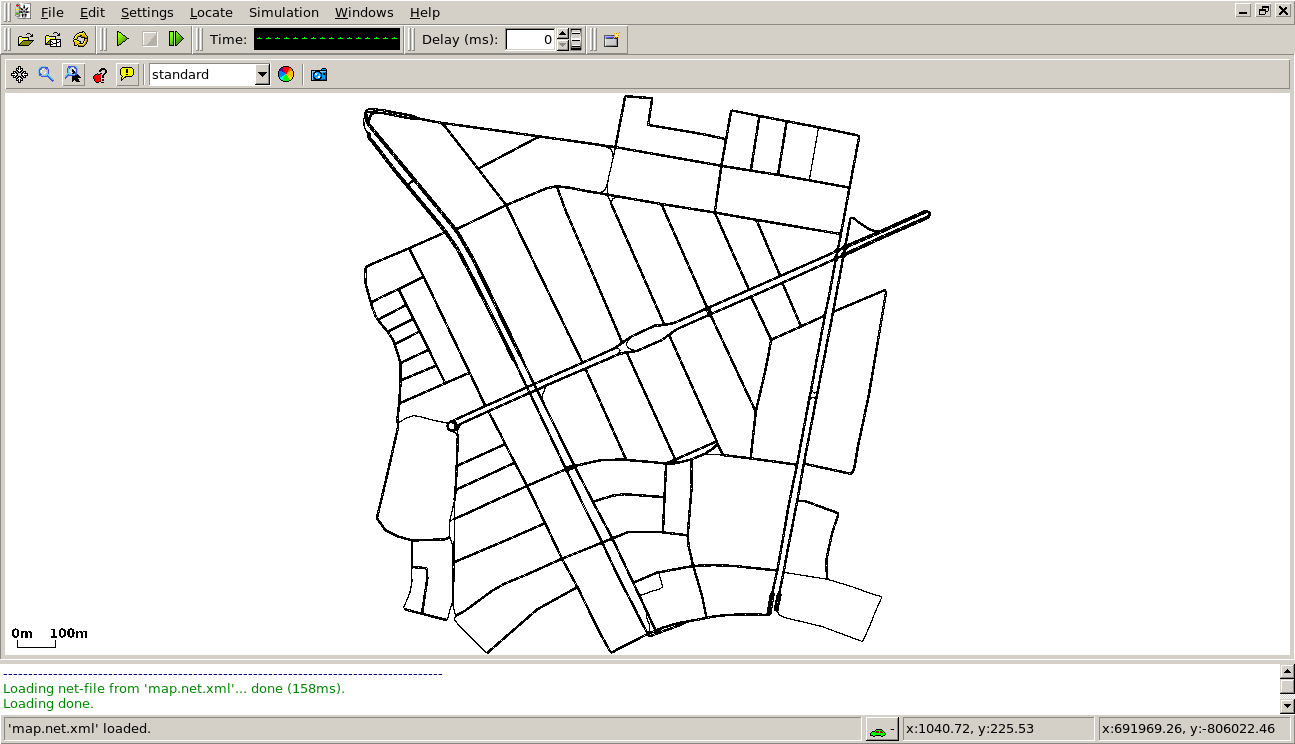
**Gambar 4.8** Ekspor Area dari OpenStreetMap

Kemudian, *file* hasil ekspor tersebut mempunyai ekstensi .osm. Setelah itu file tersebut akan dikonversi menjadi *file* .xml sama halnya dengan peta grid, namun untuk aktivitas ini menggunakan *tools* yang berbeda yaitu perintah netconvert sesuai dengan Gambar 4.9.

|  |
| --- |
| netconvert --osm-files map.osm --output-file map.net.xml |

**Gambar 4.9** Perintah dari netconvert

File .xml yang dihasilkan dapat dilihat pada Gambar 4.10 dengan menggunakan *tools* sumo-gui.



**Gambar 4.10** Hasil konversi dari Peta Real

Setelah peta dihasilkan, maka langkah selanjutnya sama seperti dengan pembuatan peta grid, yaitu melakukan konversi ke *file* .sumocfg terlebih dahulu menggunakan randomTrips dan mengkonversi *file* .sumocfg ke .tcl dengan menggunakan traceExporter.

## **4.3 Implementasi *K-Means Clustering* pada AODV**

Implementasi *K-Means Clustering* pada AODV ini dengan melakukan modifikasi terutama di bagian *control message* dan segala proses yang menggunakan *control message*. Proses *K-Means Clustering* akan dilakukan pada *node* yang ditunjuk sebagai server *node* dan dilakukan pada waktu tertentu. Clustering akan menggunakan data yakni *nodes* *positions* *table* yang terdapat pada server *node*. Pertama-tama pengisian data dari *nodes position table* ke dalam n *cluster* terlebih dahulu, setelah itu dilakukan proses penentuan *centroid* secara *random*. Potongan kode proses inisialisasi tersebut terdapat pada Gambar 4.11.

|  |
| --- |
| void AODV::runCluster(){  ......  for(int i=0;i<NUMBER\_OF\_CLUSTER;i++)  {  if(i<modClusters*Node*s)*numberNode*s = NUMBER\_OF\_NODE/NUMBER\_OF\_CLUSTER + 1;  else *numberNode*s = NUMBER\_OF\_NODE/NUMBER\_OF\_CLUSTER ;  Cluster *cluster*;  for(int j=0;j<*numberNode*s;j++)  {  Member member;  struct point pt;  pt.x = this->pt.*nodes*\_position\_x[memberId];  pt.y = this->pt.*nodes*\_position\_y[memberId];  member.setMemberPoint(pt);  member.setMemberIndex(memberId);  memberId++;  *cluster*.assignMember(member);  }  *cluster*.init();    ......  } |

**Gambar 4.11** Proses inisiasi K-means Clustering

Seluruh kode yang terdapat pada Gambar 4.11 akan ditampilkan pada lampiran A.1. Proses tersebut akan dilakukan hingga selesai lalu akan menghasilkan *node clusters table* yang berisi index *node* yang nantinya akan bertindak sebagai *cluster head* dan *cluster getaway* sesuai dengan Gambar 4.12.

|  |
| --- |
| void KMeans::run()  {  bool change = true;  while(change)  {  change = false;  for(int i=0;i<this->*clusters*.size();i++)  {  for(int j=0;j<this->*clusters*[i].members.size();j++)  {  int currentClusterIndex=i;  int nextClusterIndex=i;  int *node*Index = *clusters*[i].members[j].getMemberIndex();  for(int k=0;k<this->*clusters*.size();k++)  {  double temp = distanceMemberToCentroid(*clusters*[i].members[j],*clusters*[k].getCentroid());  if(temp<*clusters*[i].members[j].getNearestDistance())  {  *clusters*[i].members[j].setNearestDistance(temp);  nextClusterIndex = k;  }  }  if(currentClusterIndex != nextClusterIndex)  {  Member member = *clusters*[currentClusterIndex].getMember(*node*Index);  *clusters*[currentClusterIndex].unassignMember(*node*Index);  *clusters*[nextClusterIndex].assignMember(member);  change = true;  }  }  }  for(int i=0;i<this->*clusters*.size();i++)  {  this->*clusters*[i].*update*Centroid();  }  }  } |

**Gambar 4.12** Proses K-means Clustering

Seluruh kode yang terdapat pada Gambar 4.12 akan ditampilkan pada lampiran A.2. Hasil dari proses *clustering* akan ditampung pada *nodes* *clusters* *table* pada server *node*. Proses pemilihan *cluster head* dan *cluster gateway* akan dilakukan selanjutnya untuk mengetahui *node* manakah yang akan menjadi *cluster head* dan *cluster gateway*. Proses tersebut terdapat pada Gambar 4.13.

|  |
| --- |
| struct *nodes*\_*clusters*\_*table* KMeans::getResult()  {  .....  for(int i=0;i<this->*clusters*.size();i++)  {  if(*clusters*[i].getSize() != 0)ct.*nodes*\_*cluster*\_*head*[i] = *clusters*[i].getClusterHead().getMemberIndex();  }  .....  for(int i=0;i<this->*clusters*.size();i++)  {  if(*clusters*[i].getSize() != 0)ct.*nodes*\_*cluster*\_*gateway*[i] = *clusters*[i].getClusterGateway(center).getMemberIndex();  }  .....  } |

**Gambar 4.13** Proses Pemilihan Cluster Head dan Cluster Gateway

Seluruh kode yang terdapat pada Gambar 4.13 akan ditampilkan pada lampiran A.3, dan untuk kode getClusterHead() dan getClusterGateway() ada pada lampiran A.4 dan A.5.

### **4.3.1 Modifikasi Pengiriman RREQ**

Pengiriman RREQ nantinya akan disertai *nodes positions table* yang sebelumnya sudah disipsikan ke pada RREQ. Setiap *Node* akan mengirimkan RREQ sebelumnya akan mengisi informasi mengenai posisi *node* tersebut ke dalam *nodes positions table* yang ada di *node* tersebut. Potongan kode dari implementasi tersebut terdapat pada Gambar 4.14.

|  |
| --- |
| void AODV::send*Request*(nsaddr\_t dst) {  .....  ((Mobile*Node*\*) m*Node*)->getLoc(&posX,&posY);  pt.*nodes*\_position\_x[index] = posX;  pt.*nodes*\_position\_y[index] = posY;  pt.*nodes*\_*timestamp*[index] = CURRENT\_TIME;  rq->pt = this->pt;  Scheduler::instance().schedule(target\_, p, 0.);  } |

**Gambar 4.14** Modifikasi Pengiriman RREQ untuk K-Means Clustering

Seluruh kode yang terdapat pada Gambar 4.14 terdapat pada lampiran A.6.

### **4.3.2 Modifikasi Penerimaan RREQ**

Paket RREQ yang telah diterima oleh sebuah *node* akan diambil informasi mengenai *nodes positions table* yang ada pada paket tersebut, setelah itu akan dibandingkan dengan *nodes positions table* yang ada pada penerima. Perbandingan tersebut akan melihat *timestamps* atau waktu saat data tersebut didapatkan. Setelah itu *nodes positions table* yang baru akan memperbarui data *timestamps* yang ada pada paket yang telah diterima. Proses ini bertujuan agar setiap *node* yang menerima paket memiliki data yang terbaru. Potongan kode dari proses tersebut terdapat pada Gambar 4.15.

|  |
| --- |
| void AODV::recv*Request*(Packet \*p) {  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_*request* \*rq = HDR\_AODV\_REQUEST(p);  ((Mobile*Node*\*) m*Node*)->getLoc(&posX,&posY);  rq->pt.*nodes*\_position\_x[index] = this->posX;  rq->pt.*nodes*\_position\_y[index] = this->posY;  rq->pt.*nodes*\_*timestamp*[index] = CURRENT\_TIME;  for(int i=0;i<NUMBER\_OF\_NODE;i++)  {  if(this->pt.*nodes*\_*timestamp*[i] < rq->pt.*nodes*\_*timestamp*[i])  {  this->pt.*nodes*\_position\_x[i] = rq->pt.*nodes*\_position\_x[i];  this->pt.*nodes*\_position\_y[i] = rq->pt.*nodes*\_position\_y[i];  this->pt.*nodes*\_*timestamp*[i] = rq->pt.*nodes*\_*timestamp*[i];  }  }  .....  } |

**Gambar 4.15** Modifikasi Penerimaan RREQ untuk K-Means Clustering

Seluruh kode yang terdapat pada Gambar 4.15 terdapat pada lampiran A.7.

### **4.3.3 Modifikasi Pengiriman RREP**

Pengiriman RREP nantinya akan disisipkan *nodes clusters table*. Potongan kode dari proses tersebut terdapat pada Gambar 4.16

|  |
| --- |
| void AODV::send*Reply*(nsaddr\_t ipdst, u\_int32\_t hop\_count, nsaddr\_t rpdst, u\_int32\_t rpseq, u\_int32\_t lifetime, double *timestamp*) {  .....  rp->rp\_dst = rpdst;  rp->rp\_dst\_seqno = rpseq;  rp->rp\_src = index;  rp->rp\_lifetime = lifetime;  rp->rp\_*timestamp* = *timestamp*;  rp->ct = this->ct;  ......  } |

**Gambar 4.16** Modifikasi Pengiriman RREP untuk K-Means Clustering

Seluruh kode yang terdapat pada Gambar 4.16 terdapat pada lampiran A.8.

### **4.3.4 Modifikasi Penerimaan RREP**

Paket RREP yang telah diterima akan diambil informasi mengenai *nodes clusters table* setelah itu dibandingkan *node clusters table* yang ada pada penerima. Perbandingan ini akan dilakukan dengan mendasarkan *timestamps* yang ada pada *nodes clusters table*. Tabel ini akan memberikan informasi untuk suatu *node* akan bertindak sebagai *cluster* *head*, *cluster* *gateway*, ataupun member yang ada dari tiap *cluster*. Potongan kode dari proses tersebut terdapat pada Gambar 4.17.

|  |
| --- |
| void AODV::recv*Reply*(Packet \*p) {  //struct hdr\_cmn \*ch = HDR\_CMN(p);  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_reply \*rp = HDR\_AODV\_REPLY(p);  aodv\_rt\_entry \*rt;  char suppress\_reply = 0;  double delay = 0.0;  if(this->ct.*nodes*\_*cluster*\_*timestamp* < rp->ct.*nodes*\_*cluster*\_*timestamp*)  {  this->ct = rp->ct;  for(int i=0;i<NUMBER\_OF\_CLUSTER;i++)  {  if(this->ct.*nodes*\_*cluster*\_*head*[i] == this->index || this->ct.*nodes*\_*cluster*\_*gateway*[i] == this->index)  {  this->isClusterHead = true;  break;  }  else  {  this->isClusterHead = false;  }  }  }  .......  } |

**Gambar 4.17** Modifikasi Penerimaan RREP untuk K-Means Clustering

Seluruh kode yang terdapat pada Gambar 4.17 terdapat pada lampiran A.9.

### **4.3.5 Modifikasi Pengiriman Hello Messages**

Pengiriman *hello messages* akan disisipkan *nodes clusters table* dan *nodes positions table*. *Node* pengirim akan memberikan informasi posisi baru ke dalam *nodes positions table* yang ada pada *node* tersebut. Potongan kode dari proses tersebut terdapat pada Gambar 4.18.

|  |
| --- |
| void AODV::sendHello() {  Packet \*p = Packet::alloc();  struct hdr\_cmn \*ch = HDR\_CMN(p);  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_reply \*rh = HDR\_AODV\_REPLY(p);  ((Mobile*Node*\*) m*Node*)->getLoc(&posX,&posY);  this->pt.*nodes*\_position\_x[index] = posX;  this->pt.*nodes*\_position\_y[index] = posY;  this->pt.*nodes*\_*timestamp*[index] = CURRENT\_TIME;  rh->pt = this->pt;  rh->ct = this->ct;  ......  } |

**Gambar 4.17** Modifikasi Hello Messages untuk K-Means Clustering

Seluruh kode yang terdapat pada Gambar 4.18 terdapat pada lampiran A.10.

## **4.4 Implementasi algoritma *Backup Routing* pada AODV**

Implementasi dari algoritma *Backup Routing* akan dilakukan dengan modifikasi dari *control message* di bagian RREQ, RREP, RERR yang nantinya hasil modifikasi per proses tersebut akan dinamakan RRREQ, RRREP, RRERR. Namun dari ketiga perubahan tersebut memiliki perubahan yang sama, yaitu perhitungan seluruh rute sebelum mengirimkan RREQ, maka dari itu perubahan yang dilakukan adalah dari RREQ saja.

Proses dari *Backup Routing* nantinya akan menghitung seluruh proses perhitungan rute dari setiap node, yang nantinya setiap node akan menghitung seluruh jalan yang tersedia untuk mencapai node tujuan.

Proses inisiasi dari *Backup Routing* diawali dengan menambah sebuah variabel bertipe array untuk menampung energi dari setiap node dan akan menganggap seluruh *node* yang ada bukanlah sebuah *cluster head* sebelum menjalankan *K-Means Clustering* sesuai pada Gambar 4.18.

|  |
| --- |
| double avg\_energy\_node[1000];  double min\_energy\_node[1000];  MobileNode \*iNode;  xpos = 0;  ypos = 0;  iEnergy = 0.0000;  mNode = (MobileNode \*) (Node::get\_node\_by\_address(index));  isClusterHead = false; |

**Gambar 4.18** Proses Inisiasi *Backup Routing*

### **4.4.1 Modifikasi Pengiriman RREQ**

Pada saat node sumber mengirim RREQ ke pada *node* tujuan dari *node* sumber, maka akan secara otomatis menambahkan energi yang dimiliki oleh *node* sumber kepada jumlah energi pada rute lalu melakukan *reset* terhadap nilai rata-rata energi maksimum yang dimiliki oleh semua node yang ada. Seluruh kode dari proses tersebut terdapat pada Gambar 4.20.

|  |
| --- |
| iNode= (MobileNode \*) (Node::get\_node\_by\_address (index) );  xpos= iNode->X();  ypos= iNode->Y();  iEnergy= iNode->energy\_model()->energy();  for (int i=0;i<1000;i++) {  avg\_energy\_node[i] = 0;  }  rq->rq\_energy = iEnergy;  FILE \*fp;  fp = fopen("debug.txt", "a");  fprintf(fp, "\n node %d mengirim route request: energy: %f", index, rq->rq\_energy);  fclose(fp); |

**Gambar 4.19** Modifikasi Pengiriman RREQ pada *Backup Routing*

### **4.4.2 Modifikasi Penerimaan RREQ**

Setelah *node* menerima paket, setiap node akan menghitung seluruh rute yang tersedia lalu dari setiap rute rute akan dihitung jumlah energinya, dari setiap rute dipilih *node* dengan energi yang terkecil setelah itu jumlah energi pada suatu rute akan dibagi dengan *hop count* untuk mendapatkan nilai a, setelah mendapatkan nilai a, maka dipilihlah nilai a yang paling besar sesuai pada Gambar 4.20.

|  |
| --- |
| double avg\_energy, min\_energy;  rq->rq\_energy = rq->rq\_energy + iEnergy;  rq->rq\_min\_energy = iEnergy;  avg\_energy = rq->rq\_energy / (rq->rq\_hop\_count + 1);  if (rq->rq\_min\_energy < min\_energy\_node[index]) {  //Packet::free(p);  min\_energy\_node[index] = rq->rq\_min\_energy; //update minimum energi pada node  if (avg\_energy < avg\_energy\_node[index]) {  FILE \*fp;  fp = fopen("debug.txt", "a");  fprintf(fp, "\n node %d didrop: avg energi: %f", index, avg\_energy);  fclose(fp);  Packet::free(p); //drop paket jika avg energi yang diterima < max avg energi pada node  return;  }  else {  FILE \*fp;  fp = fopen("debug.txt", "a");  fprintf(fp, "\n node %d diteruskan: avg energi: %f", index, avg\_energy);  fclose(fp);  avg\_energy\_node[index] = avg\_energy; //update avg energy  }  }  else {  Packet::free(p);  return;  } |

**Gambar 4.20** Modifikasi Penerimaan RREQ pada *Backup Routing*

## **4.5 Implementasi Simulasi pada NS-2**

Implementasi untuk melakukan simulasi diawali dengan melakukan pendeskripsian dari *file* .tcl. File ini berisikan konfigurasi umum dan langkah-langkah yang dijalankan saat simulasi. Potongan kode konfigurasi dari proses tersebut terdapat pada Gambar 4.21.

|  |
| --- |
| set val(chan) Channel/WirelessChannel ;  set val(prop) Propagation/TwoRayGround ;  set val(netif) Phy/WirelessPhy ;  set val(mac) Mac/802\_11 ;  set val(ifq) Queue/DropTail/PriQueue ;  set val(ll) LL ;  set val(ant) Antenna/OmniAntenna ;  set opt(x) 700 ;  set opt(y) 700 ;  set val(ifqlen) 1000 ;  set val(nn) 200 ;  set val(seed) 1.0 ;  set val(adhoc*Routing*) AODV ;  set val(stop) 200 ;  set val(cp) "cbr.txt" ;  set val(sc) "scenario.txt" ; |

**Gambar 4.21** Konfigurasi Lingkungan Simulasi

Pada konfigurasi di Gambar 4.21, dilakukan pemanggilan *file traffic* yang berisikan konfigurasi *source* *node*, *destination* *node*, dan pengiriman paket dengan sesi, serta *file* skenario yang berisikan pergerakan *node* yang telah dibuat menggunakan SUMO sebelumnya. Seluruh kode yang terdapat pada Gambar 4.21 terdapat pada lampiran A.11.

## **4.6 Implementasi Metrik Analisis**

Implementasi untuk menguji hasil simulasi yang telah dilakukan sebelumnya akan mengukur dari beberapa metrik, yaitu PDR, E2E, HC, dan RO. Seluruh perhitungan metrik akan disatukan ke dalam *file* yang mempunyai ekstensi .awk. Seluruh kode untuk *file* .awk yang akan melakukan pengujian terdapat pada lampiran A.12.

***(Halaman ini sengaja dikosongkan)***

# BAB V UJI COBA DAN EVALUASI

Pada bab ini akan dilakukan tahap uji coba dan evaluasi sesuai dengan rancangan dan implementasi. Dari hasil uji coba yang didapatkan akan dilakukan evaluasi sehingga dapat menarik kesimpulan.

## **5.1 Lingkungan Uji Coba**

Uji coba dari Tugas Akhir ini dilakukan di perangkat dengan spesifikasi seperti pada Tabel 5.1.

|  |  |
| --- | --- |
| **Komponen** | **Spesifikasi** |
| **CPU** | Intel(R) core(TM) i7-8750H CPU @ 2.20GHz |
| **Sistem Operasi** | Linux Mint 19.3 "Tricia" - Cinnamon (64-bit) |
| **Linux Kernel** | Linux Kernel 4.4 |
| **Memori** | 16.4 GB |
| **Penyimpanan** | 50 GB |

**Tabel 5.1** Spesifikasi Perangkat Simulasi

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat dilihat pada Tabel 3.4. Pengujian dilakukan dengan melakukan simulasi menggunakan NS-2. Hasil simulasi tersebut berupa *file* dengan ekstensi .tr yang akan dianalisis dan diuji menggunakan skrip awk dan *shell script*. Ada 4 parameter yang diuji, yaitu PDR, E2E, RO, HC. Seluruh kode skrip awk terdapat pada lampiran A.12.

## **5.2 Skenario Grid**

Pengujian pada skenario *grid* digunakan untuk melihat perbandingan PDR, E2E, RO, dan AHC antara AODV asli dan AODV modifikasi. Data pada skenario ini diambil sebanyak 10 kali dalam setiap variasi kemudian hasil tersebut akan dilakukan perhitungan rata-rata untuk masing-masing variasi. *Cluster* dari setiap variasi diambil berdasarkan 10% dari jumlah *node* yang diuji. Berikut ini adalah hasil pengujian serta analisis dari keempat metriks yang diuji dalam Tugas Akhir ini.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi (%) | AODV Asli (%) | Perbedaan (%) |
| 50(5) | 86,973 | 80,368 | 6,605 |
| 100(10) | 83,139 | 61,841 | 21,298 |
| 150(15) | 77,145 | 65,476 | 11,669 |
| 200(20) | 83,295 | 69,041 | 14,254 |

**Tabel 5.2** PDR dari Hasil Simulasi Skenario *Grid*

**Gambar 5.1** Grafik PDR dari Hasil Simulasi Skenario *Grid*

Berdasarkan grafik pada Gambar 5.1, dapat dilihat bahwa dalam metriks PDR dalam setiap variasi AODV Modifikasi lebih unggul daripada AODV Asli. Perbedaan terjauh terjadi pada variasi jumlah 100 *node* dan 10 *cluster* dengan perbedaan mencapai 21,298%. Dalam grafik tersebut juga menjunjukkan bahwa performa PDR dari AODV modifikasi tidak berbeda jauh dari setiap variasi dan dapat dikatakan cukup konsisten, kecuali pada variasi jumlah *node* 150 dan jumlah *cluster* 15, PDR dari AODV Modifikasi menunjukkan angka terrendah dibandingkan dengan variasi lainnya, yaitu 77,145%, meskipun angka tersebut masih lebih tinggi daripada angka yang diperoleh AODV asli. Pada bagian ini, AODV Modifikasi menunjukkan angka yang lebih baik daripada AODV Asli dalam setiap variasi dalam skenario *grid*.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi (ms) | AODV Asli (ms) | Perbedaan (ms) |
| 50(5) | 242,34141 | 838,2864 | -595,94499 |
| 100(10) | 799,6421 | 1265,1203 | -465,4782 |
| 150(15) | 780,5188 | 962,08983 | -181,57103 |
| 200(20) | 829,46716 | 1245,8534 | -416,38624 |

**Tabel 5.3** E2E dari Hasil Simulasi Skenario *Grid*

**Gambar 5.2** Grafik E2E dari Hasil Simulasi Skenario *Grid*

Berdasarkan grafik pada Gambar 5.2, dapat dilihat bahwa dalam metriks E2E dalam setiap variasi AODV Modifikasi lebih unggul daripada AODV Asli. Sama seperti halnya PDR, dalam setiap variasi pun E2E dari AODV Modifikasi cukup konsisten, terkecuali pada variasi jumlah *node* 50 dan *cluster* 5, justru AODV Modifikasi mempunyai hasil yang amat baik, dengan waktu 242,34141 ms, AODV Modifikasi pada variasi tersebut memiliki keunggulan yang jauh dalam E2E dibandingkan dengan variasi lain. Pada bagian ini, AODV Modifikasi menunjukkan angka yang lebih baik daripada AODV Asli dalam setiap variasi dalam skenario *grid*.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi | AODV Asli | Perbedaan |
| 50(5) | 9701,1 | 1323,6 | 8377,5 |
| 100(10) | 20234,5 | 2752,1 | 17482,4 |
| 150(15) | 28704,2 | 3492,4 | 25211,8 |
| 200(20) | 37069,7 | 4537,3 | 32532,4 |

**Tabel 5.4** RO dari Hasil Simulasi Skenario *Grid*

**Gambar 5.3** Grafik RO dari Hasil Simulasi Skenario *Grid*

Berdasarkan grafik pada Gambar 5.3, dapat dilihat bahwa dalam metriks RO dalam setiap variasi AODV Modifikasi lebih buruk daripada AODV Asli. Hal itu dikarenakan RO AODV Modifikasi jauh lebih tinggi daripada AODV Asli. Terlihat juga dalam grafik tersebut, jumlah RO AODV Modifikasi dan AODV Asli meningkat seiring dengan bertambahnya jumlah *node* dan *cluster*. Namun perbedaan RO AODV Modifikasi dan AODV Asli paling banyak pada variasi jumlah *node* 200 dan jumlah *cluster* 20, dengan selisih sebanyak 32532,4. Dari hasil pengujian teratas, maka dapat disimpulkan bahwa dalam metriks RO, AODV Modifikasi lebih buruk daripada AODV Asli dalam skenario *grid*.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi | AODV Asli | Perbedaan |
| 50(5) | 3,985 | 4,078 | -0,093 |
| 100(10) | 4,178 | 5,167 | -0,989 |
| 150(15) | 4,112 | 4,253 | -0,141 |
| 200(20) | 3,757 | 4,026 | -0,269 |

**Tabel 5.5** HC dari Hasil Simulasi Skenario *Grid*

**Gambar 5.4** Grafik HC dari Hasil Simulasi Skenario *Grid*

Berdasarkan grafik pada Gambar 5.4, dapat dilihat bahwa dalam metriks HC dalam setiap variasi AODV Modifikasi lebih unggul daripada AODV Asli. Perbedaan yang terjauh terjadi pada variasi jumlah *node* 100 dan jumlah *cluster* 10, yaitu selisih 0,989. Namun untuk variasi lainnya, perbedaan HC antara AODV Modifikasi dan AODV Asli Relatif tidak berbeda jauh, namun jumlah HC yang dimiliki oleh AODV Modifikasi lebih kecil. Dalam hal HC, dapat disimpulkan bahwa AODV Modifikasi lebih unggul daripada AODV Asli dalam skenario *grid*.

## **5.3 Skenario Real**

Pengujian pada skenario *real* digunakan untuk melihat perbandingan PDR, E2E, RO, dan AHC antara AODV asli dan AODV modifikasi. Data pada skenario ini diambil sebanyak 10 kali dalam setiap variasi kemudian hasil tersebut akan dilakukan perhitungan rata-rata untuk masing-masing variasi. *Cluster* dari setiap variasi diambil berdasarkan 10% dari jumlah *node* yang diuji. Berikut ini adalah hasil pengujian serta analisis dari keempat metriks yang diuji dalam Tugas Akhir ini.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi  (%) | AODV Asli  (%) | Perbedaan  (%) |
| 50(5) | 82,608 | 77,379 | 5,229 |
| 100(10) | 78,333 | 46,266 | 32,067 |
| 150(15) | 77,269 | 53,447 | 23,822 |
| 200(20) | 71,927 | 55,834 | 16,093 |

**Tabel 5.6** PDR dari Hasil Simulasi Skenario *Real*

**Gambar 5.5** Grafik PDR dari Hasil Simulasi Skenario *Real*

Berdasarkan grafik pada Gambar 5.5, dapat dilihat bahwa dalam metriks PDR dalam setiap variasi AODV Modifikasi lebih unggul daripada AODV Asli dari simulasi tersebut. Namun pada skenario *real*, semakin banyak jumlah *node* pada simulasi, maka rasio PDR dari AODV Modifikasi semakin menurun walaupun tidak terlalu banyak jumlah penurunannya dan tetap mengungguli hasil dari AODV Asli pada skenario *real*. Perbedaan terbanyak terdapat pada variasi jumlah *node* 100 dan jumlah *cluster* 10, yaitu dengan jumlah perbedaan mencapai 32,067%. Dalam metriks PDR, dapat disimpulkan bahwa AODV Modifikasi lebih baik daripada AODV Asli dalam skenario *real*.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi (ms) | AODV Asli (ms) | Perbedaan (ms) |
| 50(5) | 471,53471 | 904,2857 | -432,75099 |
| 100(10) | 625,37918 | 1982,5671 | -1357,18792 |
| 150(15) | 711,1943 | 1848,9637 | -1137,7694 |
| 200(20) | 1111,78121 | 2676,64494 | -1564,86373 |

**Tabel 5.7** E2E dari Hasil Simulasi Skenario *Real*

**Gambar 5.6** Grafik E2E dari Hasil Simulasi Skenario *Real*

Berdasarkan grafik pada Gambar 5.6, dapat dilihat bahwa dalam metriks E2E dalam setiap variasi AODV Modifikasi lebih unggul daripada AODV Asli dari simulasi tersebut. Dalam grafik tersebut, terlihat bahwa angka dari E2E pada AODV Modifikasi bertambah seiring dengan bertambahnya jumlah *node* pada saat simulasi, namun angka dari E2E pada AODV Modifikasi masih menungguli angka dari AODV Asli. Perbedaan terbanyak terlihat pada variasi jumlah *node* 200 dan jumlah *cluster* 20, dengan perbedaan sebanyak 1564,86373 ms dengan AODV Modifikasi lebih unggul. Dari hasil pengujian diatas, maka dalam metriks E2E AODV Modifikasi lebih baik daripada AODV Asli pada skenario *real*.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi | AODV Asli | Perbedaan |
| 50(5) | 9718,8 | 1058,4 | 8660,4 |
| 100(10) | 19474,2 | 2375,4 | 17098,8 |
| 150(15) | 28210,8 | 3723,6 | 24487,2 |
| 200(20) | 35575,1 | 3988,862 | 31586,238 |

**Tabel 5.8** RO dari Hasil Simulasi Skenario *Real*

**Gambar 5.7** Grafik RO dari Hasil Simulasi Skenario *Real*

Berdasarkan grafik pada Gambar 5.7, dapat dilihat bahwa dalam metriks RO dalam setiap variasi AODV Modifikasi lebih buruk daripada AODV Asli dari simulasi tersebut. Terlihat juga perbedaan RO pada AODV Modifikasi dan AODV Asli terlampau cukup jauh, bahkan perbedaan mencapai ribuan, semakin banyak jumlah *node*, maka semakin jauh juga perbedaan RO antara AODV Modifikasi dan AODV Asli, dengan perbedaan terjauh pada jumlah *node* 200 dan jumlah *cluster* 20, dengan perbedaan mencapai 31586,238. Dalam metriks RO, dapat disimpulkan bahwa AODV Modifikasi lebih buruk daripada AODV Asli pada skenario *real*.

|  |  |  |  |
| --- | --- | --- | --- |
| Jumlah Node (Cluster) | AODV Modifikasi | AODV Asli | Perbedaan |
| 50(5) | 3,727 | 4,335 | -0,608 |
| 100(10) | 3,971 | 5,301 | -1,33 |
| 150(15) | 4,002 | 5,606 | -1,604 |
| 200(20) | 3,893 | 5,252 | -1,359 |

**Tabel 5.9** HC dari Hasil Simulasi Skenario *Real*

**Gambar 5.8** Grafik HC dari Hasil Simulasi Skenario *Real*

Berdasarkan grafik pada Gambar 5.8, dapat dilihat bahwa dalam metriks RO dalam setiap variasi AODV Modifikasi lebih baik daripada AODV Asli dari simulasi tersebut. Dengan perbedaan yang meningkat sebanding dengan jumlah *node*, namun terlihat pada variasi jumlah *node* 100 dan *cluster* 10 terdapat selisih yang paling besar dibandingkan dengan variasi lainnya, yaitu berjumlah 1,604. Jumlah HC pada AODV Modifikasi relatif konsisten, terlihat dari perbedaan yang tidak signifikan. Dalam metriks HC dapat disimpulkan bahwa AODV Modifikasi memiliki performa yang lebih baik daripada AODV Asli.

***(Halaman ini sengaja dikosongkan)***

# BAB VI KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh dari Tugas Akhir yang telah dikerjakan dan saran untuk pengembangan selanjutnya di masa yang akan datang.

## **6.1 Kesimpulan**

Kesimpulan yang diperoleh dari Tugas Akhir ini didasarkan pada hasil uji coba dan evaluasi. Kesimpulan Tugas Akhir adalah sebagai berikut:

1. Penerapan *K-Means Clustering* telah berhasil mengurangi jumlah forwarding node secara signifikan dan setelah dikombinasikan dengan algoritma *Backup Routing* secara umum menghasilkan hasil yang lebih baik daripada AODV Asli berdasarkan *Packet Delivery Ratio*, *End-to-End Delay*, dan *Average Hop Count*, namun kombinasi ini juga menyebabkan *Routing Overhead* meningkat secara fluktuatif.
2. Penerapan *K-Means Clustering* dan algoritma *Backup Routing* dalam skenario *grid* telah memengaruhi performa AODV dengan rata-rata kenaikan pada PDR sebesar 20%, rata-rata penurunan pada E2E sebesar 40%, rata-rata penurunan HC sebesar 8%, dan rata-rata kenaikkan RO sebesar 677%.
3. Penerapan *K-Means Clustering* dan algoritma *Backup Routing* dalam skenario *real* telah memengaruhi performa AODV dengan rata-rata kenaikan pada PDR sebesar 37%, rata-rata penurunan pada E2E sebesar 59%, rata-rata penurunan HC sebesar 23%, dan rata-rata kenaikkan RO sebesar 747%.

## **6.2 Saran**

Saran yang diberikan dari hasil pengujian dan analisis pada Tugas Akhir ini adalah sebagai berikut:

1. Melakukan uji coba dengan variasi yang lebih banyak sehingga mendapatkan data yang lebih akurat.
2. Menerapkan sebuah metode untuk mengurangi angka *Routing Overhead* dari AODV yang diterapkan *K-Means Clustering* dan algoritma *Backup Routing.*
3. Menerapkan metode selain *K-Means Clustering* dan algoritma *Backup Routing* yang dapat memaksimalkan kinerja dari AODV. Seperti *Particle Swarm Optimization* (PSO).

***(Halaman ini sengaja dikosongkan)***

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | M. L. Raja dan C. D. S. S. Baboo, “An Overview of MANET: Applications, Attacks and Challenges,” *International Journal of Computer Science and Mobile Computing,* vol. 3, no. 1, pp. 408-417, 2014. |
| [2] | M. Singh dan S. Kumar, “A Survey: Ad-hoc on Demand Distance Vector (AODV) Protocol,” *International Journal of Computer Applications,* vol. 161, no. 1, pp. 38-44, 2017. |
| [3] | G. G. S. Sruthy, “AODV based backup routing for optimized performance in mobile ad-hoc networks,” *2017 International Conference on Computing Methodologies and Communication (ICCMC),* pp. 684-688, 2017. |
| [4] | “OpenStreetMap,” [Online]. Available: https://www.openstreetmap.org/.. [Diakses 22 May 2020]. |
| [5] | B. A. Kumar, M. V. Subramanyam dan K. S. Prasad, “An Energy Efficient Clustering Using K-Means and AODV Routing Protocol in Ad-hoc Networks,” *International Journal of Intelligent Engineering & Systems,* vol. 12, no. 2, 2019. |
| [6] | “JOSM,” [Online]. Available: josm.openstreetmap.de/.. [Diakses 22 5 2020]. |
| [7] | K. Shaymala, S. K. Lokhande, R. B. P dan S. Kumar, “Efficient backup routing scheme in AODV with unidirectional links,” *2011 Annual IEEE India Conference,* 2011. |

# LAMPIRAN

## **A.1** Kode runCluster()

|  |
| --- |
| void AODV::runCluster(){  KMeans km;  srand(time(0));  int memberId=0;  int modClusters*Node*s = NUMBER\_OF\_NODE % NUMBER\_OF\_CLUSTER;  int *numberNode*s;  for(int i=0;i<NUMBER\_OF\_CLUSTER;i++)  {  if(i<modClusters*Node*s)*numberNode*s = NUMBER\_OF\_NODE/NUMBER\_OF\_CLUSTER + 1;  else *numberNode*s = NUMBER\_OF\_NODE/NUMBER\_OF\_CLUSTER ;  Cluster *cluster*;  for(int j=0;j<*numberNode*s;j++)  {  Member member;  struct point pt;  pt.x = this->pt.*nodes*\_position\_x[memberId];  pt.y = this->pt.*nodes*\_position\_y[memberId];  member.setMemberPoint(pt);  member.setMemberIndex(memberId);  memberId++;  *cluster*.assignMember(member);  }  *cluster*.init();  km.assignCluster(*cluster*);  }  km.run();  if(this->ct.*nodes*\_*cluster*\_*timestamp* < km.getResult().*nodes*\_*cluster*\_*timestamp*)  this->ct = km.getResult(); |

## **A.2** Kode run()

|  |
| --- |
| void KMeans::run()  {  bool change = true;  while(change)  {  change = false;  //check every member in *cluster* to each *centroid* of *clusters*  for(int i=0;i<this->*clusters*.size();i++)  {  for(int j=0;j<this->*clusters*[i].members.size();j++)  {  int currentClusterIndex=i;  int nextClusterIndex=i;  int *node*Index = *clusters*[i].members[j].getMemberIndex();  for(int k=0;k<this->*clusters*.size();k++)  {  double temp = distanceMemberToCentroid(*clusters*[i].members[j],*clusters*[k].getCentroid());  //shorter distance is found  if(temp<*clusters*[i].members[j].getNearestDistance())  {  *clusters*[i].members[j].setNearestDistance(temp);  nextClusterIndex = k;  }  }  //changing of member *cluster*  if(currentClusterIndex != nextClusterIndex)  {  Member member = *clusters*[currentClusterIndex].getMember(*node*Index);  *clusters*[currentClusterIndex].unassignMember(*node*Index);  *clusters*[nextClusterIndex].assignMember(member);  change = true;  }  }  }  //*update* *centroid*  for(int i=0;i<this->*clusters*.size();i++)  {  this->*clusters*[i].*update*Centroid();  }  }  } |

## **A.3** Kode getResult()

|  |
| --- |
| struct *nodes*\_*clusters*\_*table* KMeans::getResult()  {  struct *nodes*\_*clusters*\_*table* ct;  memset(ct.*nodes*\_*cluster*\_*head*,-1,sizeof(ct.*nodes*\_*cluster*\_*head*));  memset(ct.*nodes*\_*cluster*\_*gateway*,-1,sizeof(ct.*nodes*\_*cluster*\_*gateway*));  for(int i=0;i<this->*clusters*.size();i++)  {  if(*clusters*[i].getSize() != 0)ct.*nodes*\_*cluster*\_*head*[i] = *clusters*[i].getClusterHead().getMemberIndex();  }  struct point center;  double cX,cY;  for(int i=0;i<this->*clusters*.size();i++)  {  for(int j=0;j<this->*clusters*[i].getSize();j++)  {  cX+=*clusters*[i].members[j].getMemberPointX();  cY+=*clusters*[i].members[j].getMemberPointY();  }  }  center.x = cX/NUMBER\_OF\_NODE;  center.y = cY/NUMBER\_OF\_NODE;  for(int i=0;i<this->*clusters*.size();i++)  {  if(*clusters*[i].getSize() != 0)ct.*nodes*\_*cluster*\_*gateway*[i] = *clusters*[i].getClusterGateway(center).getMemberIndex();  }  ct.*nodes*\_*cluster*\_*timestamp* = CURRENT\_TIME;  return ct;  } |

## **A.4** Kode getClusterHead()

|  |
| --- |
| Member Cluster::getClusterHead()  {  struct point center = this->getCentroid();  double nearestDistance = 1000000.00;  int iter=0;  for(int i=0;i<this->members.size();i++)  {  double temp = distanceMemberToCentroid(members[i],center);  if(temp<nearestDistance)  {  nearestDistance=temp;  iter=i;  }  }  return members[iter];  } |

## **A.5** Kode getClusterGateway()

|  |
| --- |
| Member Cluster::getClusterGateway(struct point *clusters*Center)  {  double nearestDistance = 1000000.00;  int iter=0;  for(int i=0;i<this->members.size();i++)  {  double temp = distanceMemberToCentroid(members[i],*clusters*Center);  if(temp<nearestDistance)  {  nearestDistance=temp;  iter=i;  }  }  return members[iter];  } |

## **A.6** Kode sendRequest()

|  |
| --- |
| void  AODV::send*Request*(nsaddr\_t dst) {  // Allocate a RREQ *packet*  Packet \*p = Packet::alloc();  struct hdr\_cmn \*ch = HDR\_CMN(p);  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_*request* \*rq = HDR\_AODV\_REQUEST(p);  aodv\_rt\_entry \*rt = r*table*.rt\_lookup(dst);  //MODIFIED//  //SIGNAL  double receive*SignalStrength* = p->txinfo\_.RxPr;  double RSSM;  if(receive*SignalStrength* == 0)  {  RSSM = -200;  }  else  {  RSSM = 10\*log10(receive*SignalStrength*) + 30;  }  if(SIGNAL\_THRESHOLD >= RSSM)  {  Packet::free((Packet \*)p);  return;  }  //ENERGY  energy = m*Node*->energy\_model()->energy();  if(ENERGY\_THRESHOLD >= energy)  {  Packet::free(p);  return;  }  // MODIFIED //  assert(rt);  /\*  \* Rate limit sending of *Route* *Request*s. We are very conservative  \* about sending out *route* *request*s.  \*/  if (rt->rt\_flags == RTF\_UP) {  assert(rt->rt\_hops != INFINITY2);  Packet::free((Packet \*)p);  return;  }  if (rt->rt\_req\_timeout > CURRENT\_TIME) {  Packet::free((Packet \*)p);  return;  }  // rt\_req\_cnt is the no. of times we did network-wide *broadcast*  // RREQ\_RETRIES is the maximum *number* we will allow before  // going to a long timeout.  if (rt->rt\_req\_cnt > RREQ\_RETRIES) {  rt->rt\_req\_timeout = CURRENT\_TIME + MAX\_RREQ\_TIMEOUT;  rt->rt\_req\_cnt = 0;  Packet \*buf\_pkt;  while ((buf\_pkt = rqueue.deque(rt->rt\_dst))) {  drop(buf\_pkt, DROP\_RTR\_NO\_ROUTE);  }  Packet::free((Packet \*)p);  return;  }  #ifdef DEBUG  fprintf(stderr, "(%2d) - %2d sending *Route* *Request*, dst: %d\n",  ++*route*\_*request*, index, rt->rt\_dst);  #endif // DEBUG  // Determine the TTL to be used this time.  // Dynamic TTL evaluation - SRD  rt->rt\_req\_last\_ttl = max(rt->rt\_req\_last\_ttl,rt->rt\_last\_hop\_count);  if (0 == rt->rt\_req\_last\_ttl) {  // first time query *broadcast*  ih->ttl\_ = TTL\_START;  }  else {  // Expanding ring search.  if (rt->rt\_req\_last\_ttl < TTL\_THRESHOLD)  ih->ttl\_ = rt->rt\_req\_last\_ttl + TTL\_INCREMENT;  else {  // network-wide *broadcast*  ih->ttl\_ = NETWORK\_DIAMETER;  rt->rt\_req\_cnt += 1;  }  }  // remember the TTL used for the next time  rt->rt\_req\_last\_ttl = ih->ttl\_;  // PerHopTime is the roundtrip time per hop for *route* *request*s.  // The factor 2.0 is just to be safe .. SRD 5/22/99  // Also note that we are making timeouts to be larger if we have  // done network wide *broadcast* before.  rt->rt\_req\_timeout = 2.0 \* (double) ih->ttl\_ \* PerHopTime(rt);  if (rt->rt\_req\_cnt > 0){  rt->rt\_req\_timeout \*= rt->rt\_req\_cnt;  rt->rt\_req\_timeout += CURRENT\_TIME;  }  // Don't let the timeout to be too large, however .. SRD 6/8/99  if (rt->rt\_req\_timeout > CURRENT\_TIME + MAX\_RREQ\_TIMEOUT){  rt->rt\_req\_timeout = CURRENT\_TIME + MAX\_RREQ\_TIMEOUT;  rt->rt\_expire = 0;  }    #ifdef DEBUG  fprintf(stderr, "(%2d) - %2d sending *Route* *Request*, dst: %d, tout %f ms\n",  ++*route*\_*request*,  index, rt->rt\_dst,  rt->rt\_req\_timeout - CURRENT\_TIME);  #endif // DEBUG    // Fill out the RREQ *packet*  // ch->uid() = 0;  ch->ptype() = PT\_AODV;  ch->size() = IP\_HDR\_LEN + rq->size();  ch->iface() = -2;  ch->error() = 0;  ch->addr\_type() = NS\_AF\_NONE;  ch->prev\_hop\_ = index; // AODV hack  ih->saddr() = index;  ih->daddr() = IP\_BROADCAST;  ih->sport() = RT\_PORT;  ih->dport() = RT\_PORT;  // Fill up some more *fields*.  rq->rq\_type = AODVTYPE\_RREQ;  rq->rq\_hop\_count = 1;  rq->rq\_bcast\_id = bid++;  rq->rq\_dst = dst;  rq->rq\_dst\_seqno = (rt ? rt->rt\_seqno : 0);  rq->rq\_src = index;  seqno += 2;  assert ((seqno%2) == 0);  rq->rq\_src\_seqno = seqno;  rq->rq\_*timestamp* = CURRENT\_TIME;  // MODIFIED //  ((Mobile*Node*\*) m*Node*)->getLoc(&posX,&posY);  pt.*nodes*\_position\_x[index] = posX;  pt.*nodes*\_position\_y[index] = posY;  pt.*nodes*\_*timestamp*[index] = CURRENT\_TIME;  rq->pt = this->pt;  // MODIFIED //  Scheduler::instance().schedule(target\_, p, 0.);  } |

## **A.7** Kode recvRequest()

|  |
| --- |
| void  AODV::recv*Request*(Packet \*p) {  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_*request* \*rq = HDR\_AODV\_REQUEST(p);  // MODIFIED //  ((Mobile*Node*\*) m*Node*)->getLoc(&posX,&posY);  rq->pt.*nodes*\_position\_x[index] = this->posX;  rq->pt.*nodes*\_position\_y[index] = this->posY;  rq->pt.*nodes*\_*timestamp*[index] = CURRENT\_TIME;  for(int i=0;i<NUMBER\_OF\_NODE;i++)  {  if(this->pt.*nodes*\_*timestamp*[i] < rq->pt.*nodes*\_*timestamp*[i])  {  this->pt.*nodes*\_position\_x[i] = rq->pt.*nodes*\_position\_x[i];  this->pt.*nodes*\_position\_y[i] = rq->pt.*nodes*\_position\_y[i];  this->pt.*nodes*\_*timestamp*[i] = rq->pt.*nodes*\_*timestamp*[i];  }  else if(this->pt.*nodes*\_*timestamp*[i] > rq->pt.*nodes*\_*timestamp*[i])  {  rq->pt.*nodes*\_position\_x[i] = this->pt.*nodes*\_position\_x[i];  rq->pt.*nodes*\_position\_y[i] = this->pt.*nodes*\_position\_y[i];  rq->pt.*nodes*\_*timestamp*[i] = this->pt.*nodes*\_*timestamp*[i];  }  }  // MODIFIED //  aodv\_rt\_entry \*rt;  // MODIFIED //  //SIGNAL  double receive*SignalStrength* = p->txinfo\_.RxPr;  double RSSM;  if(receive*SignalStrength* == 0)  {  RSSM = -200;  }  else  {  RSSM = 10\*log10(receive*SignalStrength*) + 30;  }  if(SIGNAL\_THRESHOLD >= RSSM)  {  Packet::free((Packet \*)p);  return;  }  //CONGESTION  int queueLength = rqueue.queueLength(index);  if(queueLength == 0)  {  queueLength = 1;  }  //ENERGY  energy = m*Node*->energy\_model()->energy();  if(ENERGY\_THRESHOLD >= energy)  {  Packet::free(p);  return;  }  //HOPCOUNT  int hopCount = rq->rq\_hop\_count;  //PHEROMONE COUNT  double *pheromone*Count = ((RSSM+91)\*energy)/(queueLength \* hopCount);  if(rq->*pheromone*Count != NULL)  {  lastPheromoneCount = rq->*pheromone*Count + *pheromone*Count;  }  else  {  lastPheromoneCount = *pheromone*Count;  }  rq->*pheromone*Count = lastPheromoneCount;  // MODIFIED //  /\*  \* Drop if:  \* - I'm the *source*  \* - I recently heard this *request*.  \*/  if(rq->rq\_src == index) {  #ifdef DEBUG  fprintf(stderr, "%s: got my own REQUEST\n", \_\_FUNCTION\_\_);  #endif // DEBUG  Packet::free(p);  return;  }  if (id\_lookup(rq->rq\_src, rq->rq\_bcast\_id)) {  #ifdef DEBUG  fprintf(stderr, "%s: discarding *request*\n", \_\_FUNCTION\_\_);  #endif // DEBUG    Packet::free(p);  return;  }  /\*  \* Cache the *broadcast* ID  \*/  id\_insert(rq->rq\_src, rq->rq\_bcast\_id);  /\*  \* We are either going to *forward* the REQUEST or generate a  \* REPLY. Before we do anything, we make sure that the REVERSE  \* *route* is in the *route* *table*.  \*/  aodv\_rt\_entry \*rt0; // rt0 is the reverse *route*    rt0 = r*table*.rt\_lookup(rq->rq\_src);  if(rt0 == 0) { /\* if not in the *route* *table* \*/  // create an entry for the reverse *route*.  rt0 = r*table*.rt\_add(rq->rq\_src);  }    rt0->rt\_expire = max(rt0->rt\_expire, (CURRENT\_TIME + REV\_ROUTE\_LIFE));  if ( (rq->rq\_src\_seqno > rt0->rt\_seqno ) ||  ((rq->rq\_src\_seqno == rt0->rt\_seqno) &&  (rq->rq\_hop\_count < rt0->rt\_hops)) ) {  // If we have a fresher seq no. or lesser #hops for the  // same seq no., *update* the rt entry. Else don't bother.  rt\_*update*(rt0, rq->rq\_src\_seqno, rq->rq\_hop\_count, ih->saddr(), rq->*pheromone*Count,  max(rt0->rt\_expire, (CURRENT\_TIME + REV\_ROUTE\_LIFE)) );  if (rt0->rt\_req\_timeout > 0.0) {  // Reset the soft state and  // Set expiry time to CURRENT\_TIME + ACTIVE\_ROUTE\_TIMEOUT  // This is because *route* is used in the *forward* direction,  // but only *source*s get benefited by this change  rt0->rt\_req\_cnt = 0;  rt0->rt\_req\_timeout = 0.0;  rt0->rt\_req\_last\_ttl = rq->rq\_hop\_count;  rt0->rt\_expire = CURRENT\_TIME + ACTIVE\_ROUTE\_TIMEOUT;  }  /\* Find out whether any buffered *packet* can benefit from the  \* reverse *route*.  \* May need some change in the following code - Mahesh 09/11/99  \*/  assert (rt0->rt\_flags == RTF\_UP);  Packet \*buffered\_pkt;  while ((buffered\_pkt = rqueue.deque(rt0->rt\_dst))) {  if (rt0 && (rt0->rt\_flags == RTF\_UP)) {  assert(rt0->rt\_hops != INFINITY2);  *forward*(rt0, buffered\_pkt, NO\_DELAY);  }  }  }  // End for putting reverse *route* in rt *table*  // MODIFIED //  if(lastPheromoneCount != NULL)  {  rt0->*pheromone*Count = lastPheromoneCount;  }  // MODIFIED //  /\*  \* We have taken care of the reverse *route* stuff.  \* Now see whether we can send a *route* reply.  \*/  rt = r*table*.rt\_lookup(rq->rq\_dst);  // First check if I am the *destination* ..  if(rq->rq\_dst == index) {  printf("\nSAMPAI DI DESTINATION -> %d Time : %f",index,CURRENT\_TIME);  #ifdef DEBUG  fprintf(stderr, "%d - %s: *destination* sending reply\n",  index, \_\_FUNCTION\_\_);  #endif // DEBUG    // Just to be safe, I use the max. Somebody may have  // incremented the dst seqno.  seqno = max(seqno, rq->rq\_dst\_seqno)+1;  if (seqno%2) seqno++;  send*Reply*(rq->rq\_src, // IP *Destination*  1, // *Hop Count*  index, // Dest IP Address  seqno, // Dest *Sequence* Num  MY\_ROUTE\_TIMEOUT, // Lifetime  rq->rq\_*timestamp*); // *timestamp*  Packet::free(p);  }  // I am not the *destination*, but I may have a fresh enough *route*.  else if (rt && (rt->rt\_hops != INFINITY2) &&  (rt->rt\_seqno >= rq->rq\_dst\_seqno) ) {  //assert (rt->rt\_flags == RTF\_UP);  assert(rq->rq\_dst == rt->rt\_dst);  //assert ((rt->rt\_seqno%2) == 0); // is the seqno even?  if (rq->rq\_*timestamp* == NULL) rq->rq\_*timestamp* = CURRENT\_TIME;  send*Reply*(rq->rq\_src,  rt->rt\_hops + 1,  rq->rq\_dst,  rt->rt\_seqno,  (u\_int32\_t) (rt->rt\_expire - CURRENT\_TIME),  // rt->rt\_expire - CURRENT\_TIME,  rq->rq\_*timestamp*);  // Insert nexthops to RREQ *source* and RREQ *destination* in the  // precursor lists of *destination* and *source* respectively  rt->pc\_insert(rt0->rt\_nexthop); // nexthop to RREQ *source*  rt0->pc\_insert(rt->rt\_nexthop); // nexthop to RREQ *destination*  #ifdef RREQ\_GRAT\_RREP  send*Reply*(rq->rq\_dst,  rq->rq\_hop\_count,  rq->rq\_src,  rq->rq\_src\_seqno,  (u\_int32\_t) (rt->rt\_expire - CURRENT\_TIME),  // rt->rt\_expire - CURRENT\_TIME,  rq->rq\_*timestamp*);  #endif    // TODO: send grat RREP to dst if G flag set in RREQ using rq->rq\_src\_seqno, rq->rq\_hop\_counT    // DONE: Included gratuitous replies to be sent as per IETF aodv draft specification. As of now, G flag has not been dynamically used and is always set or reset in aodv-*packet*.h --- Anant Utgikar, 09/16/02.  Packet::free(p);  }  /\*  \* Can't reply. So *forward* the *Route* *Request*  \*/  else {  ih->saddr() = index;  ih->daddr() = IP\_BROADCAST;  rq->rq\_hop\_count += 1;  // Maximum *sequence* *number* seen en *route*  if (rt) rq->rq\_dst\_seqno = max(rt->rt\_seqno, rq->rq\_dst\_seqno);    // MODIFIED //  if(!isClusterHead)  {  Packet::free(p);  return;  }  // MODIFIED //  *forward*((aodv\_rt\_entry\*) 0, p, DELAY);  }  } |

## **A.8** Kode sendReply()

|  |
| --- |
| void  AODV::send*Reply*(nsaddr\_t ipdst, u\_int32\_t hop\_count, nsaddr\_t rpdst,  u\_int32\_t rpseq, u\_int32\_t lifetime, double *timestamp*) {  Packet \*p = Packet::alloc();  struct hdr\_cmn \*ch = HDR\_CMN(p);  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_reply \*rp = HDR\_AODV\_REPLY(p);  aodv\_rt\_entry \*rt = r*table*.rt\_lookup(ipdst);  printf("\n SEND REPLY BY %d to %d at %f",rpdst,ipdst,CURRENT\_TIME);  #ifdef DEBUG  fprintf(stderr, "sending *Reply* from %d at %.2f\n", index, Scheduler::instance().clock());  #endif // DEBUG  assert(rt);  rp->rp\_type = AODVTYPE\_RREP;  //rp->rp\_flags = 0x00;  rp->rp\_hop\_count = hop\_count;  rp->rp\_dst = rpdst;  rp->rp\_dst\_seqno = rpseq;  rp->rp\_src = index;  rp->rp\_lifetime = lifetime;  rp->rp\_*timestamp* = *timestamp*;  // MODIFIED //  rp->ct = this->ct;  if(rp->rp\_dst == index)  {  rp->*pheromone*Count = max(lastPheromoneCount,rt->*pheromone*Count);  }  // MODIFIED //    // ch->uid() = 0;  ch->ptype() = PT\_AODV;  ch->size() = IP\_HDR\_LEN + rp->size();  ch->iface() = -2;  ch->error() = 0;  ch->addr\_type() = NS\_AF\_INET;  ch->next\_hop\_ = rt->rt\_nexthop;  ch->prev\_hop\_ = index; // AODV hack  ch->direction() = hdr\_cmn::DOWN;  ih->saddr() = index;  ih->daddr() = ipdst;  ih->sport() = RT\_PORT;  ih->dport() = RT\_PORT;  ih->ttl\_ = NETWORK\_DIAMETER;  Scheduler::instance().schedule(target\_, p, 0.);  } |

## **A.9** Kode recvReply()

|  |
| --- |
| void  AODV::recv*Reply*(Packet \*p) {  //struct hdr\_cmn \*ch = HDR\_CMN(p);  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_reply \*rp = HDR\_AODV\_REPLY(p);  aodv\_rt\_entry \*rt;  char suppress\_reply = 0;  double delay = 0.0;  printf("\nSAMPAI KEMBALI (PERJALANAN) KE SOURCE -> %d Time : %f",index,CURRENT\_TIME);  // MODIFIED //  if(this->ct.*nodes*\_*cluster*\_*timestamp* < rp->ct.*nodes*\_*cluster*\_*timestamp*)  {  this->ct = rp->ct;  for(int i=0;i<NUMBER\_OF\_CLUSTER;i++)  {  if(this->ct.*nodes*\_*cluster*\_*head*[i] == this->index || this->ct.*nodes*\_*cluster*\_*gateway*[i] == this->index)  {  this->isClusterHead = true;  break;  }  else  {  this->isClusterHead = false;  }  }  }  // MODIFIED //  #ifdef DEBUG  fprintf(stderr, "%d - %s: received a REPLY\n", index, \_\_FUNCTION\_\_);  #endif // DEBUG  /\*  \* Got a reply. So reset the "soft state" maintained for  \* *route* *request*s in the *request* *table*. We don't *real*ly have  \* have a separate *request* *table*. It is just a part of the  \* *routing table* itself.  \*/  // Note that rp\_dst is the dest of the data *packet*s, not the  // the dest of the reply, which is the src of the data *packet*s.  rt = r*table*.rt\_lookup(rp->rp\_dst);    /\*  \* If I don't have a rt entry to this host... adding  \*/  if(rt == 0) {  rt = r*table*.rt\_add(rp->rp\_dst);  }  /\*  \* Add a *forward* *route* *table* entry... here I am following  \* Perkins-Royer AODV paper almost literally - SRD 5/99  \*/  if ( (rt->rt\_seqno < rp->rp\_dst\_seqno) || // newer *route*  ((rt->rt\_seqno == rp->rp\_dst\_seqno) &&  (rt->rt\_hops > rp->rp\_hop\_count)) ) { // shorter or better *route*    // *Update* the rt entry  rt\_*update*(rt, rp->rp\_dst\_seqno, rp->rp\_hop\_count,  rp->rp\_src, rp->*pheromone*Count, CURRENT\_TIME + rp->rp\_lifetime);  // reset the soft state  rt->rt\_req\_cnt = 0;  rt->rt\_req\_timeout = 0.0;  rt->rt\_req\_last\_ttl = rp->rp\_hop\_count;    if (ih->daddr() == index) { // If I am the original *source*  // *Update* the *route* *discovery* latency statistics  // rp->rp\_*timestamp* is the time of *request* origination  printf("\nSAMPAI KEMBALI DI SOURCE -> %d Time : %f",index,CURRENT\_TIME);  rt->rt\_disc\_latency[(unsigned char)rt->hist\_indx] = (CURRENT\_TIME - rp->rp\_*timestamp*)  / (double) rp->rp\_hop\_count;  // increment indx for next time  rt->hist\_indx = (rt->hist\_indx + 1) % MAX\_HISTORY;  }  /\*  \* Send all *packet*s queued in the sendbuffer destined for  \* this *destination*.  \* XXX - observe the "second" use of p.  \*/  Packet \*buf\_pkt;  while((buf\_pkt = rqueue.deque(rt->rt\_dst))) {  if(rt->rt\_hops != INFINITY2) {  assert (rt->rt\_flags == RTF\_UP);  // Delay them a little to help ARP. Otherwise ARP  // may drop *packet*s. -SRD 5/23/99  *forward*(rt, buf\_pkt, delay);  delay += ARP\_DELAY;  }  }  }  else {  suppress\_reply = 1;  }  /\*  \* If reply is for me, discard it.  \*/  if(ih->daddr() == index || suppress\_reply) {  Packet::free(p);  }  /\*  \* Otherwise, *forward* the *Route* *Reply*.  \*/  else {  // Find the rt entry  aodv\_rt\_entry \*rt0 = r*table*.rt\_lookup(ih->daddr());  // If the rt is up, *forward*  if(rt0 && (rt0->rt\_hops != INFINITY2)) {  assert (rt0->rt\_flags == RTF\_UP);  rp->rp\_hop\_count += 1;  rp->rp\_src = index;  *forward*(rt0, p, NO\_DELAY);  // Insert the nexthop towards the RREQ *source* to  // the precursor list of the RREQ *destination*  rt->pc\_insert(rt0->rt\_nexthop); // nexthop to RREQ *source*    }  else {  // I don't know how to *forward* .. drop the reply.  #ifdef DEBUG  fprintf(stderr, "%s: dropping *Route* *Reply*\n", \_\_FUNCTION\_\_);  #endif // DEBUG  drop(p, DROP\_RTR\_NO\_ROUTE);  }  }  } |

## **A.10** Kode sendHello()

|  |
| --- |
| void  AODV::sendHello() {  Packet \*p = Packet::alloc();  struct hdr\_cmn \*ch = HDR\_CMN(p);  struct hdr\_ip \*ih = HDR\_IP(p);  struct hdr\_aodv\_reply \*rh = HDR\_AODV\_REPLY(p);  // MODIFIED //  ((Mobile*Node*\*) m*Node*)->getLoc(&posX,&posY);  this->pt.*nodes*\_position\_x[index] = posX;  this->pt.*nodes*\_position\_y[index] = posY;  this->pt.*nodes*\_*timestamp*[index] = CURRENT\_TIME;  rh->pt = this->pt;  rh->ct = this->ct;  // MODIFIED //  #ifdef DEBUG  fprintf(stderr, "Sending Hello from %d at %.2f\n", index, Scheduler::instance().clock());  #endif // DEBUG  rh->rp\_type = AODVTYPE\_HELLO;  //rh->rp\_flags = 0x00;  rh->rp\_hop\_count = 1;  rh->rp\_dst = index;  rh->rp\_dst\_seqno = seqno;  rh->rp\_lifetime = (1 + ALLOWED\_HELLO\_LOSS) \* HELLO\_INTERVAL;  // ch->uid() = 0;  ch->ptype() = PT\_AODV;  ch->size() = IP\_HDR\_LEN + rh->size();  ch->iface() = -2;  ch->error() = 0;  ch->addr\_type() = NS\_AF\_NONE;  ch->prev\_hop\_ = index; // AODV hack  ih->saddr() = index;  ih->daddr() = IP\_BROADCAST;  ih->sport() = RT\_PORT;  ih->dport() = RT\_PORT;  ih->ttl\_ = 1;  Scheduler::instance().schedule(target\_, p, 0.0);  } |

## **A.11** Kode konfigurasi skenario NS-2

|  |
| --- |
| # ======================================================================  # Define options  # ======================================================================  set val(chan) Channel/WirelessChannel ;# channel type  set val(prop) Propagation/TwoRayGround ;# radio-propagation model  set val(netif) Phy/WirelessPhy ;# network interface type  set val(mac) Mac/802\_11 ;# MAC type  set val(ifq) Queue/DropTail/PriQueue ;# interface queue type  set val(ll) LL ;# link layer type  set val(ant) Antenna/OmniAntenna ;# antenna model  set opt(x) 700 ;# X dimension of the topography  set opt(y) 700 ;# Y dimension of the topography  set val(ifqlen) 1000 ;# max packet in ifq  set val(nn) 50 ;# how many nodes are simulated  set val(seed) 1.0 ;  set val(adhocRouting) AODV ;# routing protocol  set val(stop) 200 ;# simulation time  set val(cp) "cbr.txt" ;#<-- traffic file  set val(sc) "scenario.txt" ;#<-- mobility file  set val(energy\_mod) EnergyModel ;# energy model  set val(energy\_init) 100 ;# init val for energy  set val(tx\_power) 1.65 ;# energy consume for transmiting packet  set val(rx\_power) 1.4 ;# energy consume for receiving packet  set val(idle\_power) 0.5 ;# energy consume for idle  set val(sleep\_power) 0.3 ;# energy consume for sleep mode  # =====================================================================  # Main Program  # ======================================================================  # Initialize Global Variables  # create simulator instance  set ns\_ [new Simulator]  # setup topography object  set topo [new Topography]  set tracefd [open scenario.tr w]  set namtrace [open scenario.nam w]  $ns\_ trace-all $tracefd  $ns\_ namtrace-all-wireless $namtrace $opt(x) $opt(y)  # set up topology object  set topo [new Topography]  $topo load\_flatgrid $opt(x) $opt(y)  # Create God  set god\_ [create-god $val(nn)]  $ns\_ node-config -adhocRouting $val(adhocRouting) \  -llType $val(ll) \  -macType $val(mac) \  -ifqType $val(ifq) \  -ifqLen $val(ifqlen) \  -antType $val(ant) \  -propType $val(prop) \  -phyType $val(netif) \  -channelType $val(chan) \  -energyModel $val(energy\_mod) \  -initialEnergy $val(energy\_init) \  -txPower $val(tx\_power) \  -rxPower $val(rx\_power) \  -idlePower $val(idle\_power) \  -sleepPower $val(sleep\_power) \  -topoInstance $topo \  -agentTrace ON \  -routerTrace ON \  -macTrace ON \  -movementTrace ON    ###  # 802.11p default parameters  Phy/WirelessPhy set RXThresh\_ 3.65262e-10 ; #250m  Phy/WirelessPhy set CSThresh\_ 3.65262e-10 ; #250m  ###  # Create the specified number of nodes [$val(nn)] and "attach" them  # to the channel.  for {set i 0} {$i < $val(nn)} {incr i} {  set node\_($i) [$ns\_ node]  $node\_($i) random-motion 0 ;# disable random motion  }  # Define node movement model  puts "Loading connection pattern..."  source $val(cp)  # Define traffic model  puts "Loading scenario file..."  source $val(sc)  # Define node initial position in nam  for {set i 0} {$i < $val(nn)} {incr i} {  # 20 defines the node size in nam, must adjust it according to your scenario  # The function must be called after mobility model is defined    $ns\_ initial\_node\_pos $node\_($i) 20  }  # Tell nodes when the simulation ends  for {set i 0} {$i < $val(nn) } {incr i} {  $ns\_ at $val(stop).0 "$node\_($i) reset";  }  #$ns\_ at $val(stop) "stop"  $ns\_ at $val(stop).0002 "puts \"NS EXITING...\" ; $ns\_ halt"  puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y $opt(y) rp $val(adhocRouting)"  puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $val(seed)"  puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"  puts "Starting Simulation..."  $ns\_ run |

## **A.12** Kode konfigurasi awk

|  |
| --- |
| BEGIN {  sendLine = 0;  recvLine = 0;  fowardLine = 0;  TC =0;  rt\_pkts=0;  rt\_send=0;  rt\_forward=0;  recvd = 0;  hc = 0;  }  $0 ~/^s.\* AGT/ {  sendLine ++ ;  }  $0 ~/^r.\* AGT/ {  recvLine ++ ;  }  $0 ~/^f.\* RTR/ {  fowardLine ++ ;  }  $0 ~/^s.\* \[TC / {  TC ++ ;  }  {  if (( $1 == "r") && ( $7 == "cbr" || $7 =="tcp" ) && ( $4=="AGT" )) recvd++;  if (( $1 == "r") && ( $4 == "RTR") && ( $7 == "cbr")) { hc = hc + 1; }  }  {  if($4 == "AGT" && $1 == "s" && seqno < $6) {  seqno = $6;  }  #end-to-end delay  if($4 == "AGT" && $1 == "s") {  start\_time[$6] = $2;  } else if(($7 == "cbr") && ($1 == "r")) {  end\_time[$6] = $2;  } else if($1 == "D" && $7 == "cbr") {  end\_time[$6] = -1;  } else if (($1 == "s" || $1 == "f") && ($4 == "RTR") && ($7 == "AODV")) {    rt\_pkts++;  }  if (($1 == "s") && ($4 == "RTR") && ($7 == "AODV") && ($25 == "(REQUEST)")) {    rt\_send++;  }  if (($1 == "s") && ($4 == "RTR") && ($7 == "AODV") && ($25 == "(REQUEST)") && ($3 != "\_58\_")) {    rt\_forward++;  }    }  END {    for(i=0; i<=seqno; i++) {  if(end\_time[i] > 0) {  delay[i] = end\_time[i] - start\_time[i];  count++;  }  else  {  delay[i] = -1;  }  }  for(i=0; i<=seqno; i++) {  if(delay[i] > 0) {  n\_to\_n\_delay = n\_to\_n\_delay + delay[i];  }  }  n\_to\_n\_delay = n\_to\_n\_delay/count;  printf "Packet sendLine \t= %d \n", sendLine;  printf "Packet recvLine \t= %d \n", recvLine;  printf "Packet PDR Ratio \t= %.4f \n", (recvLine/sendLine);  printf "Packet loss \t= %d \n", (sendLine-recvLine);  printf "Packet forwardLine\t= %d \n", fowardLine;  printf "End-to-End Delay \t= " n\_to\_n\_delay \* 1000 " ms \n";  printf "Topology Control \t= %d \n", TC;  printf "Routing Packets \t= %d \n", rt\_pkts;  printf "Route Request Send \t= %d \n", rt\_send;  printf "Route Request Forwarded \t= %d \n", rt\_forward;  printf sendLine"\t"recvLine"\t%.4f\t"(sendLine-recvLine)"\t" n\_to\_n\_delay \* 1000" ms\t" rt\_pkts"\t"rt\_send"\t" rt\_forward"\n",(recvLine/sendLine);  printf("AHC : %.2f\n",hc/recvd);  } |

# BIODATA PENULIS



**FAHRIZAL NAUFAL AHMAD**, lahir di Kediri, 8 Agustus 1998. Penulis merupakan anak sulung dari tiga bersaudara. Penulis menempuh Pendidikan sekolah dasar di SDI Al-Azhar 20 Cibubur, kemudian melanjutkan sekolah di SMP Labschool Jakarta, setelah itu menempuh Pendidikan sekolah menengah atas di SMA Labschool Jakarta, setelah lulus dari Pendidikan SMA, Penulis melanjutkan Pendidikan Sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember, Surabaya.

Dalam menempuh pendidikan sarjana, Penulis mengambil bidang minat Arsitektur Jaringan dan Komputer (AJK). Penulis juga aktif dalam bersosialisasi dan berorganisasi selama menempuh pendidikan di kampus perjuangan. Penulis mengikuti beberapa organisasi, seperti HMTC ITS, dan menjadi Kepala Departemen Hubungan Luar BEM FTIK ITS. Penulis juga sempat berkegiatan di UKM CLICK ITS selama awal perkuliahan. Selain itu, penulis juga sempat menjadi penanggung jawab dari REEVA SCHEMATICS 2018, salah satu sub-event dari SCHEMATICS 2018. Penulis juga sempat melakukan *internship* di PT. SIGMA CIPTA CARAKA atau yang biasa kita kenal sebagai TELKOMSIGMA. Jika ada pertanyaan atau saran, silakan menghubungi penulis di +6281320465964 atau fahrizal.8898@hotmail.com.