TESIS TE-142599

PERANCANGAN GEOGRAPHIC INFORMATION SYSTEM UNTUK TRACKING DAN ROUTING MENGGUNAKAN ALGORITMA DIJKSTRA PADA KENDARAAN ANGKUTAN UMUM

MUH. ARISTO INDRAJAYA
2213206201

DOSEN PEMBIMBING
Dr. Ir. Achmad Affandi, DEA
Dr. Istas Pratomo ST, MT.

PROGRAM MAGISTER
JURUSAN TEKNIK ELEKTRO
BIDANG KEAHLIAN TELEMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPERMBER
SURABAYA
2015

THESIS TE-142599

DESIGN OF GEOGRAPHIC INFORMATION SYSTEM FOR TRACKING AND ROUTING USING DIJKSTRA ALGORITHM ON PUBLIC TRANSPORTATION

MUH. ARISTO INDRAJAYA
2213206201

SUPERVISOR
Dr. Ir. Achmad Affandi, DEA
Dr. Istas Pratomo ST, MT.

POST GRADUATE PROGRAM
DEPARTMENT OF ELECTRICAL ENGINEERING
AREAS OF EXPERTISE TELEMATICS
FACULTY OF INDUSTRIAL OF TECHNOLOGY
SEPULUH NOPEMBER OF TECHNOLOGY
SURABAYA
2015

Thesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
Di
Institut Teknologi Sepuluh Nopember

Oleh :
Muh. Aristo Indrajaya
NRP : 2213206201
Tanggal Ujian : 7 Januari 2016
Periode Wisuda : Maret 2016

Disetujui oleh:

1. Dr. Ir. Achmad Affandi, DEA (Pembimbing I)
   NIP. 196510141990021001

2. Dr. Istas Pratomo, ST, MT. (Pembimbing II)
   NIP. 197903252003121001

3. Dr. Adhi Dharma Wibawa, ST., MT. (Penguji)
   NIP. 19760505 200812 1003

4. Dr. Ir. Wirawan, DEA (Penguji)
   NIP. 19631102 198903 1 011

5. Eko Setijadi, ST., MT., Ph.D. (Penguji)
   NIP. 19721001 200312 1 002

6. Dr. Surya Sumpeno, ST., M.Sc. (Penguji)
   NIP. 196906131997021003

Direktur Program Pasca Sarjana,

Prof. Ir. Djauhar Manfaat. M.Sc, Ph.D
NIP. 19601202 198701 1 001

PERANCANGAN GEOGRAPHIC INFORMATION SYSTEM UNTUK TRACKING DAN ROUTING MENGGUNAKAN ALGORITMA DIJKSTRA PADA KENDARAAN ANGKUTAN UMUM

Nama Mahasiswa	: Muh. Aristo Indrajaya
NRP	: 2213206201
Pembimbing	: Dr. Ir. Achmad Affandi, DEA.
Pembimbing II	: Dr. Istas Pratomo, ST, MT.

ABSTRAK

Kemampuan untuk melakukan pelacakan (*tracking*) dan perutean (routing) sebuah kendaraan bermotor sangat berguna dalam kehidupan sehari – hari, seperti pengamanan pada kendaraan pribadi, sistem transportasi publik, menejemen armada transportasi masal dan lainnya. Pada penelitian ini, kami merancang sebuah aplikasi *Geographic Information System* (GIS) yang akan diterapkan pada layanan angkutan umum. Sistem ini akan secara *real time* akan melakukan *tracking* terhadap posisi tiap kendaraan (taksi) serta mampu melakukan pelayanan otomatis terhadap setiap permintaan taksi oleh pelanggan.

Algoritma Dijkstra adalah algoritma pencarian graf yang memecahkan masalah jalur terpendek yang bersumber dari satu simpul untuk sebuah graf dengan bobot simpul tidak boleh negatif. Analisis dilakukan dengan cara memeriksa simpul dengan bobot terkecil dan memasukkannya ke dalam himpunan solusi dengan awal pencarian simpul asal membutuhkan pengetahuan tentang semua jalur dan bobotnya. Algoritma Dijkstra yang diterapkan pada sistem ini akan berfungsi menemukan taksi paling layak bagi pelanggan dengan menggunakan paremeter jarak dan tingkat kepadatan lalu-lintas sebagai nilai bobotnya.

Kata Kunci : *GPS, GIS, GSM, Dijkstra, TCP/IP*

DESIGN OF GEOGRAPHIC INFORMATION SYSTEM FOR TRACKING AND ROUTING USING DIJKSTRA ALGORITHM ON PUBLIC TRANSPORT VEHICLE

Name : Muh. Aristo Indrajaya
NRP : 2213206201
Advisor : Dr. Ir. Achmad Affandi, DEA.
Dr. Istas Pratomo, ST, MT.

ABSTRACT

The ability to perform tracking and routing of a motor vehicle is very useful in daily life - today, as security in personal vehicles, public transportation systems, fleet management and other mass transportation. In this study, we designed an application of Geographic Information System (GIS) that will be applied to public transport services. This system will be in real time tracking of the position of each vehicle (taxi) and is able to perform automated service to every request a taxi by passengers.

Dijkstra's algorithm is a graph search algorithm that solves the shortest path problem that originates from one node to a graph with node weights can not be negative. The analysis was performed by examining the node with the smallest weight and put it into the set of solutions to the initial search origin node requires knowledge of all the lines and weight. Dijkstra's algorithm is applied to this system will work to find a taxi most feasible for customers using the parameter range and level of traffic density as weight values.

Keywords : *GPS, GIS, GSM, Dijkstra, TCP/IP*

KATA PENGANTAR

Dengan Nama Allah yang Maha Pengasih lagi Maha Penyayang

Segala puja dan puji syukur kepada Allah SWT atas segala rahmat dan karunia yang telah dilimpahkan kepada penulis sehingga penulisan thesis dengan judul :

PERANCANGAN GEOGRAPHIC INFORMATION SYSTEM UNTUK TRACKING DAN ROUTING MENGGUNAKAN ALGORITMA DIJKSTRA PADA KENDARAAN ANGKUTAN UMUM

Dapat diselesaikan dengan baik. Buku thesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar magister pada program studi teknik elektro dengan bidang keahlian Telematika, Institute Teknologi Sepuluh Nopember.

Pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih sedalam-dalamnya kepada :

1. Orang tuaku, ayahanda Hasan Basri dan ibunda Syamsiar tercinta yang telah mendidik dan merawat penulis sampai bisa berada pada posisi ini.
2. Bapak Achmad Affandi dan Istas Pratomo atas bimbingan serta kesabarannya terhadap penulis sehingga bisa menyelesaikan thesis ini tepat pada waktunya.
3. Bapak dan Ibu dosen Telematika yang telah memberi banyak pengetahuan baru bagi penulis selama masa perkuliahan.
4. Sofyan Saputra, Heru Purnomo Kurniawan, dan Wildan Bakaramah serta Kanda Adrin Thamrin yang telah banyak membantu dalam proses penelitian ini.
5. Mba Dyna Indar Karunia Putri atas bantuan dan gagasannya dalam membangun sistem ini.
6. Kepada guruku Rendra Towidjojo yang sudah mengajarkan banyak hal padaku mulai dari masalah teknis tentang jaringan sampai filosofi hidup termasuk bahwa betah tinggal di zona aman adalah sebuah kesalahan besar.
7. Teman-teman angkatan 2008 S1 Teknik Elektro Universitas Tadulako atas segala dukungannya.

8. Teman-teman di UPT. TIK Universitas Tadulako atas segala bantuan dan dukungannya.
9. Rekan-rekan S2 dan S1 di lab B301 atas kebaikan serta kerjasamanya selama penulis menempuh studi pascasarja di Jurusan Teknik elektro.
10. Rekan-rekan di STMIK Adhi Guna Palu yang telah banyak membantu dalam pengujian sistem yang dibangun.

Penulis menyadari bahwa dalam penulisan thesis ini masih jauh dari kata sempurna, untuk itu demi perbaikan dan penyempurnaan thesis ini maka saran dan kritik membangun sangat diharapkan. Besar harapan penulis bahwa buku thesis ini dapat member informasi dan manfaat bagi pembaca pada umumnya dan mahasiswa jurusan teknik elektro pada khususnya.

Surabaya, 27 Desember 2015

Penulis

Muh. Aristo Indrajaya

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pertumbuhan jumlah kendaraan secara global diperkirakan akan semakin meningkat seiring dengan pertumbuhan ekonomi serta jumlah masyarakat kelas menengah yang semakin meningkat, seperti di Cina dan India. Meskipun pertumbuhan jumlah kendaraan selain memberi dampak positif yaitu semakin bertambahnya jumlah penduduk kelas menengah keatas, akan tetapi pengembangan sistem pelacakan (*tracking*) dan perutean (routing) suatu kendaraan yang juga merupakan bagian yang sangat penting pada sebuah sistem transportasi masih sangat kurang. Penggunaan sistem *tracking* dan *routing* pada mode transportasi akan sangat bermanfaat untuk berbagai macam aspek termasuk aspek keamanan maupun ekonomi pada kendaraan pribadi, sistem transportasi masal, kendaraan niaga dan lain sebagainya. Untuk angkutan penumpang dan barang, sistem *tracking* dan *routing* ini dapat digunakan untuk melakukan menejemen terhadap armada angkutan khususnya untuk pengaturan terhadap rute yang akan dilalui sehingga akan meningkatkan efisiensi terhadap waktu tempuh dan pemakaian bahan bakar yang harganya terus mengalami lonjakan.

Di sisi lain, sistem navigasi GPS (*Global Positioning System*) telah diadopsi secara luas sebagai sistem navigasi kendaraan saat ini dan harganya kian terjangkau. Perangkat ini digunakan oleh pengemudi untuk membantu menemukan tujuan mereka melalui instruksi – instruksi. Sistem GPS ini juga akan dapat digunakan untuk melakukan *tracking* terhadap posisi suatu kendaraan berdasarkan pembacaan koordinatnya dan dengan menganalisa data dari GPS, kecepatan sebuah kendaraan juga dapat diketahui dengan pasti. Dengan membangun aplikasi GIS (*Geographic Information System*) berbasis *web*, maka posisi dan kecepatan suatu kendaraan dapat diketahui melalui perangkat *mobile* yang terhubung ke internet.

Tujuan utama dari penelitian kami adalah membangun sebuah sistem yang mampu melacak posisi suatu kendaraan secara otomatis (*automatic vehicle location tracking system*) yang dapat menunjang kinerja angkutan penumpang khususnya armada taksi. Pada umumnya penyedia layanan taksi masih menggunakan metode manual dengan mengandalkan pemancar radio untuk memberikan informasi mengenai pemesanan taksi kepada pengendara yang dilakukan oleh seorang operator. Hal ini menyebabkan belum adanya konfirmasi taksi mana yang akan melayani pemesanan taksi nantinya serta memperpanjang waktu pelayanan utamanya di jam-jam sibuk. Pencarian taksi terdekat untuk pemesan taksi terdekat juga tidak bisa dilakukan tanpa mengetahui lokasi pemesan taksi secara otomatis.

Dengan menggunakan sistem ini, selain posisi setiap taksi akan terpantau melalui sebuah aplikasi GIS (*Geographic Information System*) berbasis *web*, sistem ini akan mampu melakukan pemilihan taksi yang paling layak diberikan kepada pelanggan berdasarkan data posisi kendaraan dan tingkat pendapatan dari tiap pengendara taksi sehingga akan memberikan keuntungan bukan hanya bagi calon penumpan saja tapi juga bagi pengendara itu sendiri yang akan berdampak pada pemerataan pendapatan bagi tiap – tiap pengendara taksi.

1.2 Rumusan Masalah

Belum adanya suatu sistem yang mampu melakukan *tracking* terhadap posisi kendaraan secara akurat dan otomatis menyediakan kendaraan dengan jarak yang terdekat dari pelanggan yang dapat diterapkan pada transportasi angkutan umum khususnya taksi menyebabkan pelanggan sering kali harus menunggu terlalu lama untuk mendapatkan taksi yang telah dipesan terutama pada jam kerja. Hal ini diakibatkan sistem layanan yang ada masih mengandalkan layanan komunikasi radio untuk menginformasikan adanya pemesanan taksi sehingga proses yang ada masih dikerjakan secara konvensional oleh seseorang yang bertindak sebagai operator.

1.3 Batasan Masalah

a. Pengujian ini akan dilakukan pada jaringan komunikasi seluler berbasis GSM.

b. Pengujian ini akan menggunakan perangkat GPS berbasis Android.
c. Sistem yang dibangun diakses dengan menggunakan aplikasi berbasis *web browser*.
d. Parameter yang diukur adalah kecepatan layanan dari sistem yang dibangun.
e. Pengujian akan dilakukan pada wilayah kota Palu, Sulawesi Tengah.

1.4 Tujuan

Tujuan utama dari penelitian ini adalah membangun sebuah sistem yang mampu melakukan *tracking* terhadap posisi dan kecepatan suatu kendaraan angkutan umum menggunakan GPS dan GIS yang dapat menampilkannya secara *online* melalui aplikasi berbasis *web mapping* serta melakukan proses *routing* memanfaatkan Algoritma Dijkstra yang memungkinkan sistem yang dibangun untuk mencari taksi dengan posisi terdekat dan tingkat kepadatan yang lancar dari pelanggan dalam waktu yang cepat.

1.5 Manfaat

a. Sebagai acuan dalam pengembangan sistem *tracking* pada sistem transportasi cerdas (*intelegence transportation system*) khususnya pada aplikasi *tracking* dan *routing* kendaraan.
b. Memberikan kontribusi pada pengembangan teknologi navigasi khususnya bagi pihak yang berkecimpung pada bidang sistem transportasi cerdas.
c. Meningkatkan pelayanan kepada konsumen khususnya pada sistem transportasi angkutan umum serta meningkatkan eifisiensi bagi operator transportasi.

1.6 Metodologi

a. Studi literatur, yaitu dengan yaitu dengan membaca dan memahami teori-teori yang berkaitan dengan penelitian yang dilakukan yang terdiri dari buku-buku referensi dan jurnal – jurnal terakreditasi.
b. Eksperimen, yaitu dengan merancang aplikasi *tracking* yang berfungsi melacak posisi dan kecepatan suatu kendaraan serta merancang aplikasi berbasis *web* yang berfungsi menampilkan data posisi dan kecepatan suatu kendaraan.

c. Analisis, yaitu pengujian kelayakan terhadap sistem *tracking* yang dirancang.
d. Komparasi, yaitu membandingkan kemampuan dari sistem yang dibangun sekarang dengan sistem yang pernah dibangun sebelumnya.

BAB II
KAJIAN PUSTAKA

2.1 Kajian Pustaka

Penelitian mengenai sistem *tracking* terhadap posisi dan kendaraan secara otomatis (*automatic vehicle location tracking system*) telah banyak dilakukan oleh berbagai peneliti dari berbagai negara antara lain :

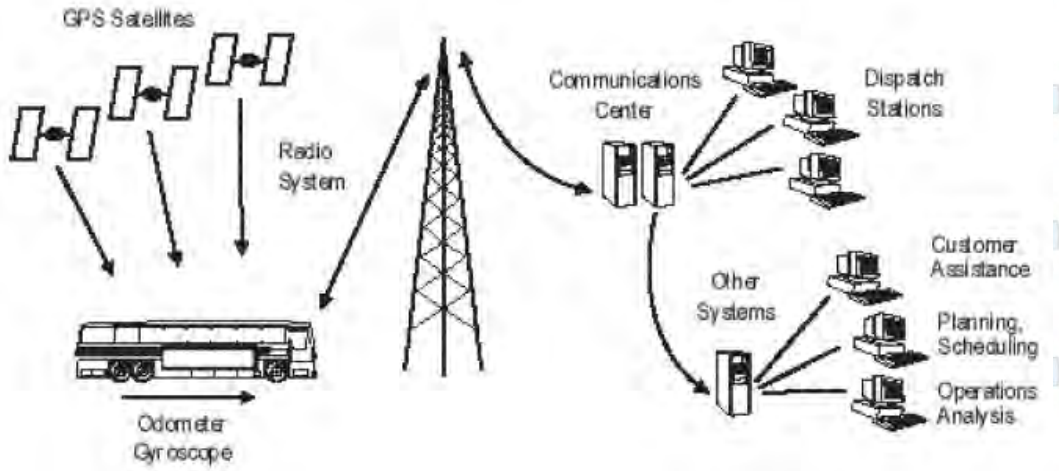
1. Aloquili (2009) dalam penelitiannya telah merancang sebuah sistem yang mampu melakukan pelacakan (*tracking*) pada sebuah kendaraan dan menampilkan lokasi dari kendaraan tersebut pada sebuah aplikasi sistem informasi geografis (GIS). Sistem ini diterapkan pada armada angkutan barang dimana dengan menerapkan algoritma dijkstra pada sistem ini, rute yang akan ditempuh kendaraan barang menuju tujuan dapat direncanakan seefisien mungkin.
2. Menard (2011) telah merancang sebuah sistem yang mampu merancang sebuah sistem yang mampu melakukan *tracking* terhadap setiap kendaraan dengan penggunaan perangkat *GPS receiver* berbasis *smartphone*. Penelitian ini lebih bertujuan pada pengujian keandalan terhadap beberapa perangkat *smartphone* dalam melakukan *tracking*. Dari penelitian yang dilakuakan dapat disimpulkan bahwa perangkat *smartphone* iPhone 4 dan Motorola Droid X memiliki kemampuan *tracking* yang lebih baik dibandingkan perangkat *smartphone* lainnya.
3. Gintoro (2010) dalam penelitiannya telah mengembangkan sistem yang mampu melacak dan mencari taksi dengan jarak paling dekat dari penumpang yang melakukan pemesanan dengan menggunakan Metode Haversine.
4. Dat (2013) dalam penelitiannya telah merancang sebuah sistem yang mampu melakukan *tracking* terhadap kendaraan dan dapat mengirimkan informasi mengenai posisi kendaraan melalui layanan SMS. Sistem ini dibangun dengan tujuan meningkatkan keamanan sebuah kendaraan dari tindakan pencurian.

5. Lee (2014) dalam penelitiannya telah merancang sebuah sistem yang mampu melacak sebuah kendaraan secara *realtime* dimana informasi lokasi dari tiap kendaraan dikirimkan melalui layanan GPRS/GSM menuju sebuah *server*. Layanan Google Maps API yang terintegrasi pada sistem ini akan menampilkan posisi dari tiap-tiap kendaraan melalui sebuah aplikasi berbasis *web*.

2.2 Landasan Teori

2.2.1 Automatic Vehicle Location (AVL)

Automatic vehicle location (AVL) adalah sistem pelacak (*tracking*) kendaraan kendaraan berbasis komputer [8]. Posisi terkini setiap kendaraan akan termonitor dan akan diteruskan menuju pusat kendali (*control center*). Umumnya, informasi posisi kendaraan akan disimpan pada kendaraan untuk yang dapat berlangsung selama beberapa detik. Informasi posisi dapat diteruskan menuju pusat kendali dalam bentuk data mentah atau setelah diproses oleh perangkat yang ada pada kendaraan sebelum ditransmisikan.



Gambar 1.1 Prinsip kerja AVL

Agen atau perusahaan transportasi telah menerapkan sistem AVL ini untuk membantu mereka dalam banyak hal. Dengan menerapkan sistem ini maka banyak manfaat yang dapat diambil seperti darii sisi operasional, sistem ini akan membantu memenuhi ketepatan jadwal, meningkatkan efisiensi layanan, fasisilitas sistem yang terintegrasi serta mengurangi jumlah pengawas (*supervisor*) di lapangan. Adapun manfaat yang dapat diambil dari segi komunikasi maupun

kemanan adalah menggantikan sistem radio yang telah tua, mengurangi komunikasi suara yang melalui terminal data *mobile* serta meningkatkan waktu respon terhadap insiden dan keadaan darurat.

Teknologi yang telah banyak digunakan saat ini untuk melacak kendaraan pada sistem AVL adalah GPS (*Global Position System*). Teknologi GPS menggunakan sinyal yang ditransmisikan dari jaringan 24 satelit yang mengorbit mengelilingi bumi.

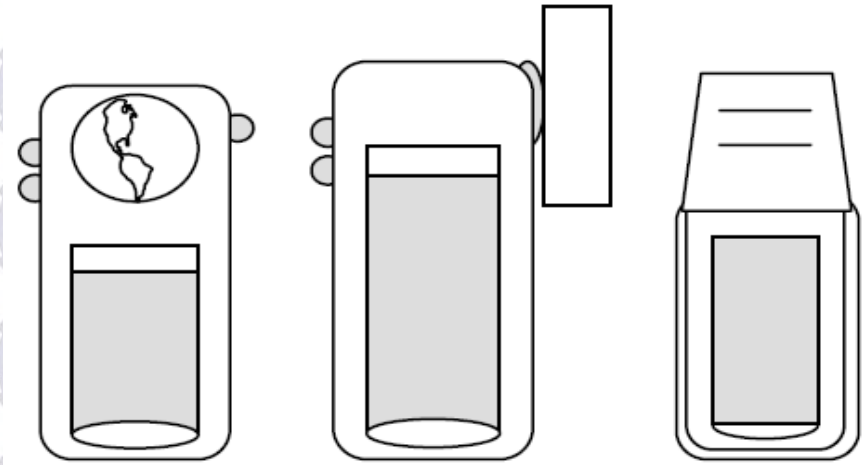
Dua metode yang umum digunakan dalam proses transmisi data pada sistem AVL adalah melalui metode *polling* dan *exception reporting* melalui jaringan *wireless* [8]. Melalui *polling*, komputer melakukan pengambilan terhadap data posisi setiap kendaraan. Metode ini mewajibkan setiap kendaraan untuk dapat membaca dan memperhitungkan posisinya. Posisi kendaraan kemudian akan ditransmisikan menuju pusat kendali. Waktu yang dibutuhkan untuk melakukan pengambilan data (*polling*) akan semakin bertambah jika jumlah kendaraan juga bertambah. Akan tetapi, karena komputer mampu melakukan pengambilan data terhadap beberapa kendaraan secara simultan melalui beberapa kanal radio yang berbeda maka waktu yang dibutuhkan untuk melakukan satu kali siklus *polling* bertumpu pada jumlah kanal radio yang dimanfaatkan. Pada metode *exception reporting*, setiap kendaraan melaporkan posisinya kepada pusat hanya pada beberapa lokasi tertentu atau ketika kendaraan berjalan tidak sesuai dengan jadwal yang ditentukan. Penggunaan metode ini akan lebih menghemat penggunaan kanal radio yang digunakan.

Data yang telah ditransmisikan oleh kendaraan akan dikumpulkan oleh perangkat *Computer Assisted Dispatching* (CAD) yang akan menunjukkan status kendaraan, kondisi, posisi, jadwal, serta informasi akan insiden yang terjadi pada komputer pusat. CAD juga mengatur komunikasi dan membantu operator untuk mengambilkeputusan dengan tepat berdasarkan data yang ditampilkan serta mengumpulkan data yang dibutuhkan oleh sebuah agen transportasi.

2.2.2 Global Positioning System (GPS)

GPS merupakan sistem yang berfungsi menentukan letak di permukaan bumi dengan bantuan penyelarasan oleh satelit [12]. GPS menggunakan data dari satelit untuk mengkalkulasi keakuratan posisi di bumi. Semua GPS bekerja

dengan cara yang sama tetapi mereka terlihat sangat berbeda dan memiliki *software* yang berbeda pula serta memiliki bentuk yang beraneka ragam, tergantung dari *vendor* pembuatnya. Model dari tiap GPS dapat dilihat pada Gambar 2.2. Perbedaan yang sangat signifikan di antara berbagai GPS *receiver* adalah jumlah satelit yang secara simultan dapat berkomunikasi dengan *receiver* tersebut. Kebanyakan *receiver* dideskripsikan dengan 12 *channel* yang berarti perangkat tersebut mampu berkomunikasi dengan 12 satelit. GPS tipe terdahulu hanya mampu 8 atau 5 *channel*, akan tetapi dengan model terbaru *receiver* mampu berkomunikasi dengan 14 – 20 satelit [8].



Gambar 1.2 Model Perangkat GPS

Jumlah, posisi dan kekuatan sinyal dari satelit memungkinkan GPS untuk menghitung tingkat kesalahan yang terjadi. Tingkat kesalahan ini dapat menjadi panduan yang baik untuk mengetahui seberapa akurat pembacaan yang dilakukan oleh GPS dan tingkat kesalahan yang dapat ditolerir dari perangkat GPS adalah di bawah 10 m (idealnya dibawah 5 m).

GPS merekam data umumnya sama disemua unit. GPS *receiver* secara otomatis merekam data kedalam memorinya berdasarkan waktu yang berlalu atau jarak dari pergerakan yang terjadi. Hal ini umumnya diistilahkan sebagai *trackpoints*.

GPS *receivers* dapat seringkali dpat digunakan sebagai alat navigasi yang lengkap, tidak hanya menawarkan petunjuk arah dan lokasi secara mendetail tetapi juga menawarkan alat navigasi yang mampu membantu pengguna ketika

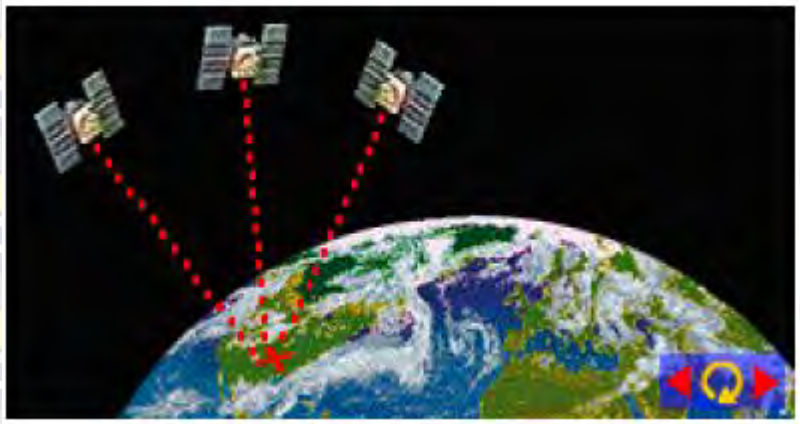
akan berpindah dari satu tempat ke tempat lain. Kebanyakan *receivers* telah dilengkapi dengan kompas digital. Kompas digital bekerja berdasarkan data dari satelit dan ini bukanlah merupakan kompas yang terbuat dari magnet serta hanya akan bekerja ketika pengguna mulai bergerak.

2.2.2.1 Prinsip Kerja GPS

Setiap daerah di atas permukaan bumi ini minimal terjangkau oleh 3-4 satelit. Pada prakteknya, setiap GPS terbaru bisa menerima sampai dengan 12 *channel* satelit sekaligus. Kondisi langit yang cerah dan bebas dari halangan membuat GPS dapat dengan mudah menangkap sinyal yang dikirimkan oleh satelit. Semakin banyak satelit yang diterima oleh GPS, maka akurasi yang diberikan juga akan semakin tinggi. Cara kerja GPS secara logika adalah sebagai berikut :

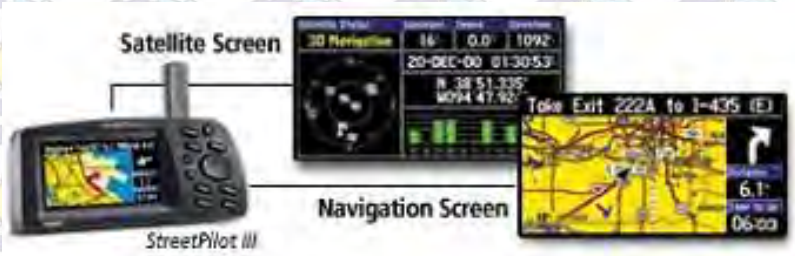
a. Memakai perhitungan "*triangulation*" dari satelit.
b. Untuk perhitungan "*triangulation*", GPS mengukur jarak menggunakan travel time sinyal radio.
c. Untuk mengukur travel time, GPS memerlukan memerlukan akurasi waktu yang tinggi.
d. Untuk perhitungan jarak, kita harus tahu dengan pasti posisi satelit dan ketingian pada orbitnya.
e. Terakhir harus menggoreksi delay sinyal waktu perjalanan di atmosfer sampai diterima reciever.

Satelit GPS berputar mengelilingi bumi selama 12 jam di dalam orbit yang akurat dia dan mengirimkan sinyal informasi ke bumi. GPS reciever mengambl informasi itu dan dengan menggunakan perhitungan "*triangulation*" menghitung lokasi *user* dengan tepat [8] berdasarkan data yang diterima oleh satelit seperti yang dapat dilihat pada Gambar 2.3. GPS *receiver* membandingkan waktu sinyal di kiirim dengan waktu sinyal tersebut di terima. Dari informasi itu didapat diketahui berapa jarak satelit. Dengan perhitungan jarak jarak GPS reciever dapat melakukan perhitungan dan menentukan posisi user dan menampilkan dalam peta elektronik.



Gambar 1.3 Pengiriman data oleh satelit menuju perangkat *receiver*

Sebuah GPS reciever harus mengunci sinyal minimal tiga satelit untuk memenghitung posisi 2D (*latitude* dan *longitude*) dan track pergerakan. Jika GPS reciever dapat menerima empat atau lebih satelit, maka dapat menghitung posisi 3D (*latitude*, *longitude* dan *altitude*) [8]. Jika sudah dapat menentukan posisi *user*, selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah yang dituju, jalur, tujuan perjalanan, jarak tujuan, matahari terbit dan matahari terbenam dan masih banyak lagi.



Gambar 1.4 Tampilan GPS *receiver*

Satelit GPS dalam mengirim informasi waktu sangat presesi karena Satelit tersebut memakai jam atom. Jam atom yang ada pada satelit jalam dengan partikel atom yang di isolasi, sehingga dapat menghasilkan jam yang akurat dibandingkan dengan jam biasa.

Perhitungan waktu yang akurat sangat menentukan akurasi perhitungan untuk menentukan informasi lokasi kita. Selain itu semakin banyak sinyal satelit yang dapat diterima maka akan semakin presesi data yang diterima karena ketiga satelit mengirim pseudo-random code dan waktu yang sama.

Satelit harus tetap pada posisi yang tepat sehingga stasiun di bumi harus terus memonitor setiap pergerakan satelit, dengan bantuan radar yang presesi salalu di cek tentang altitude, posision dan kecepatannya.

Sinyal yang dikirimkan oleh satelit ke GPS akan digunakan untuk menghitung waktu perjalanan (travel time). Waktu perjalanan ini sering juga disebut sebagai *Time of Arrival* (TOA). Sesuai dengan prinsip fisika, bahwa untuk mengukur jarak dapat diperoleh dari waktu dikalikan dengan cepat rambat sinyal.

Maka, jarak antara satelit dengan GPS juga dapat diperoleh dari prinsip fisika tersebut. Setiap sinyal yang dikirimkan oleh satelit akan juga berisi informasi yang sangat detail, seperti orbit satelit, waktu, dan hambatan di atmosfir. Satelit menggunakan jam atom yang merupakan satuan waktu paling presisi.

Untuk dapat menentukan posisi dari sebuah GPS secara dua dimensi (jarak), dibutuhkan minimal tiga buah satelit. Empat buah satelit akan dibutuhkan agar didapatkan lokasi ketinggian (secara tiga dimensi). Setiap satelit akan memancarkan sinyal yang akan diterima oleh GPS *receiver*. Sinyal ini akan dibutuhkan untuk menghitung jarak dari masingmasing satelit ke GPS. Dari jarak tersebut, akan diperoleh jari-jari lingkaran jangkauan setiap satelit. Lewat perhitungan matematika yang cukup rumit, interseksi (perpotongan) setiap lingkaran jangkauan satelit tadi akan dapat digunakan untuk menentukan lokasi dari GPS di permukaan bumi.

2.2.2.2 Model dan Interkoneksi GPS

Sebuah GPS juga memiliki firmware yang bisa di-upgrade. Upgrade firmware ini biasanya disediakan pada site produsen GPS tersebut. Upgrade firmware biasanya menggunakan kabel yang dibundel atau-pun tersedia sebagai asesoris. Kabel ini juga ternyata bisa digunakan untuk menghubungkan GPS ke komputer (baik itu notebook, PC, maupun PDA dengan sedikit bantuan konverter). Software GPS yang tersedia untuk berbagai platform tersebut juga cukup banyak. Dengan software tersebut, pengguna dapat dengan mudah mendownload informasi dari GPS. Memori sebuah GPS memang relatif terbatas, sehingga kemampuan ekstra untuk menyimpan informasi yang pernah ditempuh

oleh pengguna ke PC/PDA (yang biasanya memiliki memori lebih besar) tentu akan sangat menyenangkan. Untuk media komunikasi GPS dengan *hardware* lain selain kabel, model GPS sekarang juga ada yang dilengkapi dengan Bluetooth, Infrared.

Berdasarkan fisik, model GPS dibagi menjadi beberapa tipe antara lain model portable/handheld (ukurannya menyerupai ponsel), ada yang lebih besar (biasanya dimount di mobil/kapal), ada pula yang meng-gunakan interface khusus untuk dikoneksikan ke notebook maupun PDA (Palm, Pocket PC maupun Nokia Com-municator).

GPS untuk keperluan *outdoor* biasanya juga dilengkapi dengan perlindungan anti air dan tahan ben-turan. Beberapa GPS keluaran terakhir bahkan sudah menyediakan layar warna dan kemampuan komunikasi radio jarak pendek (FRS/Family Radio Service).

Jika suatu saat Anda ingin pergi ke lokasi yang pernah Anda kunjungi dengan menggunakan GPS. Maka, Anda tinggal mengunggah data yang pernah Anda simpan di komputer kembali ke GPS. Selanjutnya, Anda akan mendapatkan rekaman perjalanan Anda terdahulu. Lokasi dan track yang pernah Anda kunjungi akan dapat Anda temui kembali dengan cepat, dan tentu saja meminimalkan resiko tersesat.

2.2.3 Geographic Information System (GIS)

Pertama kali Sistem Informasi Geografi digunakan secara nasioanal adalah di Kanada sekitar tahun 1960, oleh *Canada Geographic Information System* (CGIS) dalam proyek untuk pengembangan kemampuan lahan nasional (*National land capability*) dengan cara mengkompilasi dan inventarisasi potensi lahan produktif di Kanada [4]. Beberapa tahun sejak proyek CGIS Canada tersebut, GIS mulai intensif dikembangkan di berbagai bagian dunia khususnya di Eropa dan Amerika, bahkan badan dunia FAO (*Food and Agriculture Organization*) mulai intensif menggunakan SIG sejak tahun 1970.

GIS awalnya berkembang dari dua independent disiplin ilmu yaitu : kartografi diijital dan database. Perkembangan dalam kartografi djital sebagai hasil dari berkembangnya dunia desain khususnya CAD (*Computer Aided Design*) sejak tahun 1960an. Demikian pula perkembangan penggunaan data base

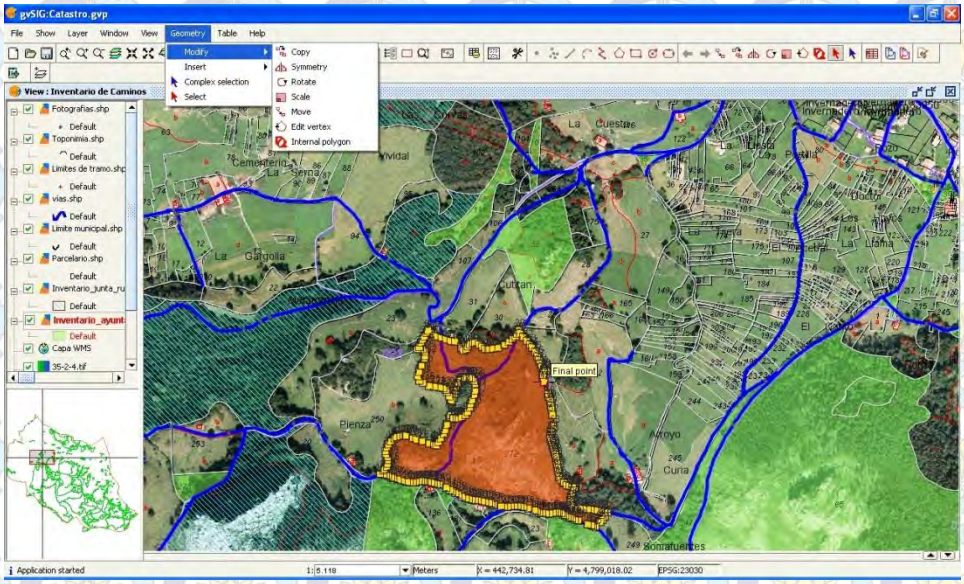
khusunya sistem pengelolaan database atau *Data Base Management Systems* (DBMS) yang memungkinkan integrasi data spasial dan non-spasial turut andil dalam mempercepat perkembangan SIG. Dalam Perkembangan lanjut GIS melibatkan berbagai disiplin yang sebenarnya saat ini menjadi akar dari perkembangan kedepan seperti remote sensing, fotogrametri dan survei.

Model data CAD (*Computer Aided Design*) adalah sistem komputer pertama yang dapat menggambar peta dengan garis – garis yang ditampilkan pada tabung sinar katoda dan peta raster menggunakan karakter tertentu pada printer garis. Model CAD menawarkan perbaikan atau perangkat keras grafis dan perangkat lunak untuk pemetaan yang bisa menghasilkan peta dengan kaidah kartografi tinggi.

Sejak 1981, ESRI (*Environmental System Research Institute*) Inc. Mengenalkan model data berorientasi objek dikenal *data coverage* dan model database georasional, yang merek dagang komersialnya dikenal sebagai arcinfo. Selanjutnya sekitar tahun 2000 dikenalkan generasi lanjut berupa model data *geodatabase*, atau dikenal komersial sebagai arcView / arcGIS.

Dalam model *coverge* dan *geodatabase atribute* di simpan dalam tabel data yang saling berhubungan. Konsep data tabel meliputi : (1) Tabel terdiri atas *row* dan *column*, (2) Tiap baris mempunyai *column* yang sama, (3) Tiap *column* mempunyai tipe data khusus seperti *integer*, desimal, *character* dan penanggalan, dan (4) Fungsi operasi relasional dapat dijalankan.

Saat ini penggunaan GIS telah sangat terintegrasi dengan kehidupan sehari – hari. Teknologi geospasial yang bekerja dibelakang layar telah membantu baik secara langsung atau tidak langsung berbagai proyek dan program pemerintah. Fitur peta saat ini tersedia hampir disetiap sistem informasi. Mulai dari pelayanan informasi (*service*) hingga kebutuhan pencarian alamat, sesuatu hal yang sangat sulit terjadi padaera tahun 1980-an di Indonesia.



Gambar 1.5 Sebuah aplikasi GIS

2.2.3.1 Komponen Utama GIS

GIS merupakan sistem yang mampu memetakan sebuah lokasi melalui pemetaan berbasis komputer. GIS merupakan sistem yang terdiri dari beberapa perangkat yang saling mendukung satu sama lain yaitu :

a. Perangkat keras (*Hardware*)

GIS membutuhkan komputer untuk penyimpanan dan pemproresan data. Ukuran dari sistem komputerisasi bergantung pada tipe GIS itu sendiri. GIS dengan skala yang kecil hanya membutuhkan PC *(personal computer)* yang kecil dan sebaliknya. Ketika GIS yang di buat berskala besar di perlukan spesifikasi komputer yang besar pula serta *host* untuk *client machine* yang mendukung penggunaan *multiple user*. Hal tersebut disebabkan data yang digunakan dalam GIS baik data vektor maupun data raster penyimpanannya membutuhkan ruang yang besar dan dalam proses analisanya membutuhkan memori yang besar dan prosesor yang cepat. Untuk mengubah peta ke dalam bentuk digital diperlukan *hardware* yang disebut *digitizer*.

b. Perangkat lunak (*Software*)

Dalam pembuatan GIS di perlukan *software* yang menyediakan fungsi *tool* yang mampu melakukan penyimpanan data, analisis dan menampilkan

informasi geografis. Dengan demikian, elemen yang harus terdapat dalam komponen software GIS adalah:

1. Tool untuk melakukan input dan transformasi data geografis
2. Sistem Manajemen Basis Data (DBMS)
3. Tool yang mendukung query geografis, analisa dan visualisasi
4. *Graphical User Interface (GUI)* untuk memudahkan akses pada tool geografi

Inti dari software GIS adalah software GIS itu sendiri yang mampu menyediakan fungsi-fungsi untuk penyimpanan, pengaturan, *link*, *query* dan analisa data geografi. Beberapa contoh software GIS adalah ArcView, MapInfo, ArcInfo untuk SIG; CAD system untuk *entry graphic data*; dan ERDAS serta ER-MAP untuk proses *remote sensing* data. Modul dasar perangkat lunak SIG: modul pemasukan dan pembetulan data, modul penyimpanan dan pengorganisasian data, modul pemrosesan dan penyajian data, modul transformasi data, modul interaksi dengan pengguna (*input query*).

c. Data

Sumber data yang dapat digunakan dalam masukan data antara lain data pengindraan jauh, data teristris, dan data peta.

1. Data Pengindraan Jauh

   Data pengindraan jauh berupa citra, baik citra foto maupun nonfoto. Apabila sumber data berupa foto udara, harus diolah terlebih dahulu dengan cara interpretasi, kemudian disajikan dalam bentuk peta. Namun apabila berupa citra satelit yang sudah dalam bentuk digital dapat langsung digunakan setelah dilakukan koreksi seperlunya.

2. Data *Teristris*

   Data *teristris* adalah data yang diperoleh langsung dari pengukuran lapangan, antara lain pH tanah, salinitas air, curah hujan, dan persebaran penduduk. Data teristris dapat disajikan dalam bentuk peta, tabel, grafik, atau hasil perhitungan saja.

3. Data Peta

Data peta adalah data yang sudah dalam bentuk peta yang siap digunakan. Guna keperluan GIS melalui komputerisasi, data-data dalam peta dikonversikan ke dalam bentuk digital. Sebuah peta harus benar-benar mempresentasikan sebagian atau seluruh permukaan bumi. Oleh karena itu, sebuah peta harus memenuhi syarat-syarat berikut ini:.

- Jarak antartitik pada peta harus sesuai dengan jarak antartitik sesungguhnya di permukaan bumi.
- Luas wilayah pada peta harus sesuai dengan luas wilayah sesungguhnya.
- Sudut atau arah sebuah garis pada peta harus sesuai dengan sudut arau arah yang sesungguhnya di permukaan bumi.
- Bentuk sebuah objek pada peta harus sesuai dengan bentuk yang sesungguhnya di permukaan bumi

2.2.3.2 Tahapan Pemrosesan Data Pada GIS

Tahapan pemrosesan data pada GIS meliputi tiga tahapan berikut yaitu proses pemasukan data, pmanipulasi dan analisis data serta proses menampilkan data.

a. *Proses pemasukan data*

Proses pemasukan data ke dalam GIS diawali dengan mengumpulkan dan menyiapkan data spasial maupun data atribut dari berbagai sumber data, baik yang bersumber dari data lapangan, peta, penginderaan jauh, maupun data statistik. Bentuk data yang akan dimasukkan dapat berupa tabel, peta, catatan statistik, laporan, citra satelit, foto udara, dan hasil survei atau pengukuran lapa ngan. Data tersebut diubah terlebih dahulu menjadi format data digital sehingga dapat diterima sebagai masukan data yang akan disimpan ke dalam GIS. Data yang masuk ke dalam GIS dinamakan database (data dasar atau basis data). Dari digitasi peta dihasilkan layer peta tematik. Layer peta tematik adalah peta yang digambar pada sesuatu yang bersifat tembus pandang, seperti plastik transparan. Berbagai fenomena di permukaan bumi dapat dipetakan ke dalam beberapa layer peta tematik, dengan setiap layernya merupakan

representasi kumpulan benda (feature) yang memiliki kesamaan. Misalnya, layer jalan, kemiringan lereng, daerah aliran sungai, tata guna lahan, dan jenis tanah. Layer-layer ini kemudian disatukan dan disesuaikan urutan maupun skalanya. Kemampuan ini memungkinkan seseorang untuk mencari di mana letak suatu daerah, objek, atau hal lainnya di permukaan bumi. Fungsi ini dapat digunakan, seperti untuk mencari lokasi rumah, mencari rute jalan, dan mencari tempat-tempat penting yang ada di peta. Pengguna GIS dapat pula melihat pola-pola yang mungkin akan muncul dengan melihat penyebaran letak feature, seperti sekolah, sungai, jembatan, dan daerah pertambangan. Teknik pemasukan data pada GIS dapatmenggunakan cara berikut :

1) *Digitasi data-data spasial, seperti peta dengan menggunakan digitizer.*
2) *Scaning data-data spasial dan atribut dengan menggunakan scanner.*
3) *Modifikasi data terutama data atribut.*
4) *Mentransfer data-data digital, seperti citra satelit secara langsung.*

b. Manipulasi dan analisis data.

Tahapan manipulasi dan analisis data adalah tahapan dalam GIS yang berfungsi menyimpan, menimbun, menarik kembali, memanipulasi, dan menganalisis data yang telah tersimpan dalam komputer. Beberapa macam analisis data, antara lain sebagai berikut.

1) *Analisis lebar, yaitu analisis yang dapat menghasilkan gambaran daerah tepian sungai dengan lebar tertentu. Kegunaannya antara lain untuk perencanaan pembangunan jembatan dan bendungan, seperti bendungan Jatiluhur, Saguling, dan Cirata yang mem bendung Citarum.*
2) *Analisis penjumlahan aritmatika, yaitu analisis yang dapat menghasilkan peta dengan klasifikasi baru. Kegunaannya antara lain untuk perencanaan wilayah, seperti wilayah permukiman, industri, konservasi, dan pertanian.*

3) *Analisis garis dan bidang, yaitu analisis yang digunakan untuk menentukan wilayah dalam radius tertentu. Kegunaannya antara lain untuk menentukan daerah rawan bencana, seperti daerah rawan banjir, daerah rawan gempa, dan daerah rawan gunung api.*

c. Keluaran data

Tahapan keluaran data, yaitu tahapan dalam GIS yang berfungsi menyajikan atau menampilkan hasil akhir dari proses GIS dalam bentuk peta, grafik, tabel, laporan, dan bentuk informasi digital lainnya yang diperlu kan untuk perencanaan, analisis, dan penentuan kebijakan terhadap suatu objek geografis. Misalnya, untuk mendukung pengambilan keputusan dalam perencanaan dan pengelolaan penggunaan lahan (land use), sumber daya alam, lingkungan, transportasi, fasilitas kota, dan pelayanan umum lainnya. Kemampuan inilah yang membedakan GIS dengan sistem informasi lainnya yang membuatnya menjadi berguna untuk berbagai kalangan dalam menjelaskan kejadian, merencanakan strategi, dan memprediksi apa yang akan terjadi.

2.2.4 Transmission Control Protocol / Internet Protocol (TCP/IP)

TCP/IP merupakan sebuah kumpulan besar protokol yang memungkinkan komputer untukmelakukan komunikasi. TCP/IP mendefinisikan secara terperinci protokol – protokol yang terdapat dalam TCP/IP kedalam RFC (*Request for Comments*) [7]. Melalui penggunaan protokol – protokol yangterdapat dalam TCP/IP RFC, sebuah komputer dapat berkomunikasi dengan komputer lain yang juga mengimplementasikan TCP/IP.

Seperti arsitektur jaringan lainnya, TCP/IP mengelompokkan beberapa protokol kedalam lapisan – lapisan (*layers*) yang berbeda yang dapat dilihat pada tabel 2.1.

Tabel 1.1 Model arsitektur protokol TCP/IP dan contoh protokolnya

Lapisan Arsitektur TCP/IP	Contoh Protokol
Applicarion	HTTP, POP3, SMTP
Transport	TCP,UDP

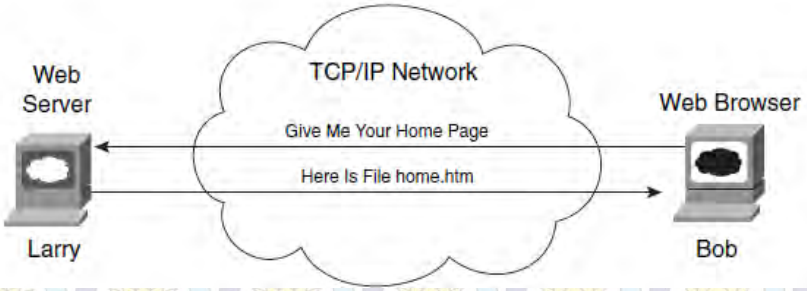
Internet	IP
Network access	*Ethernet, Frame Relay*

Jika sebuah aplikasi yang baru ingin dibuat, protokol yang digunakan oleh aplikasi akan dianggap sebagai protokol lapisan aplikasi (*application layer*). Sebagai contoh, ketika WWW (*World Wide Web*) pertama kali dibuat, sebuah lapisan protokol aplikasi yang baru telah dibuat untuk tujuan permintaan halaman *web* dan penerimaan konten – konten dari halaman web. Begitu pula dengan yang terjadi pada lapisan *network access* beserta protokol dan standarnya seperti Ethernet. Jika seseorang ingin membangun sebuah model LAN terbaru, protokol yang akan digunakan akan dianggap sebagai sebuah bagian dari lapisan *network access*.

2.2.4.1 TCP/IP Application *Layer*

Protokol yang bekerja pada lapisan aplikasi (*application layer*) menyediakan layanan bagi aplikasi perangkat lunak yang berjalan pada sebuah komputer. Lapisan aplikasi bukan merupakan aplikasi itu sendiri, akan tetapi lapisan aplikasi menyediakan layanan yang dibutuhkan oleh sebuah perngkat lunak seperti kemampuan untuk melakukan pengiriman sebuah *file* pada sebuah aplikasi HTTP. Secara singkat, lapisan aplikasi menyediakan layanan antarmuka antara perangkat lunak yang berjalan pada sebuah komputer dengan jaringan itu sendiri.

Saat ini, salah satu aplikasi yang populer dari TCP/IP saat ini adalah *web browser*. Mayoritas pembuat (*vendor*) perangkat lunak telah melakukan modifikasi pada perangkat lunak mereka untuk mendukung akses dari *web browser*. Berkat hal tersebut maka penggunaan *web browser* sangatlah mudah. Pengguna dapat mengakses sebuah halaman *web* dengan mengetikkan nama dari sebuah *website* dan halaman *web* tersebut akan terbuka. Untuk memahami proses yang berjalan ketika seorang *client* melakukan *request* terhadap sebuah halaman *web* serta proses *reply* yang dilakukan oleh *web server* dapat dilihat pada gambar 3.5. berikut.



Gambar 1.6 Proses *request* sebuah halaman *web*

Pada gambar 3.5. diatas, *web browser* dari *client* (Bob) akan melakukan *request* halaman *web* kepada *web server* (Larry). Aplikasi *web server* yang dimiliki oleh Larry telah dikonfigurasi untuk mengetahui bahwa sebuah halaman *web* secara *default* terdapat didalam sebuah *file* yang bernama home.htm. Bob menerima *file* dari Larry dan menampilkannya pada halaman *web browser*. Pada contoh kasus ini digunakan dua protokol lapisan aplikasi. Pertama adalah proses *request* untuk *file* dan proses transfer *file* dibangun berdasarkan protokol HTTP (*Hypertext Transfer Protocol*).

Protokol lain yang digunakan adalah HTML (*Hypertext Markup Language*). HTML adalah salah satu darisedemikian banyak spesifikasi yang menjelaskan bagaimana *web browser* dari *client* harus menafsirkan *text* didalam sebuah *file* yang diterima. Secara secerhana, sebuah *file* mungkin saja mengandung petunjuk mengenai sebuah *text* seperti ukuran, warna, dan sebagainya. Dalam banyak kasus, sebuah *file* juga mengandung petunjuk mengenai *file* lain yang harus didapatkan oleh *web browser client* seperti gambar atau animasi. Protokol HTTP akan digunakan untuk mendapatkan beberpa *file* tersebut dari *web server*. Untuk dapat memahami prosesnya secara lebih jelas, dapat dilihat pada gambar 3.6. di bawah ini.

Gambar 1.7 Proses HTTP *request* dan HTTP *response*

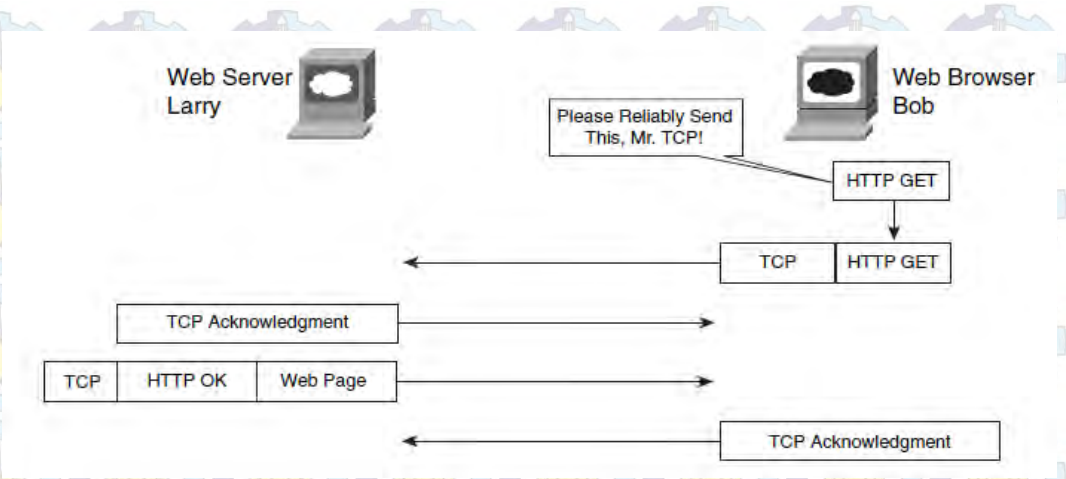
Untuk mendapatkan halaman *web* dari *server* (Larry), *client* (Bob) mengirimkan sesuatu yang disebut sebagai HTTP *header* menuju *server*. *Header* ini mengandung perintah untuk mendapatkan *file*. Paket *request* secara khusus mengandung nama *file* (home.htm dalam kasus ini) atau jika tidak terdapat nama *file* yang disebutkan, *web server* akan mengasumsikan bahwa *client* menginginkan halaman *web* standar.

Respons yang dilakukan oleh *server* adalah dengan mengirimkan HTTP *header*. Informasi yang terkandung dalam *header* mengindikasikan bahwa *request* yang dilakukan oleh *client* dapat dilayani. Misalnya, Jika *client* mencari halaman *web* yang tidak ditemukan, dan hanya menerima pesan "HTTP 404 *not found*", itu berarti *client* menerima kode balasan HTTP 404. Jika halaman *web* yang diinginkan dapat ditemukan, kode balasan yang dikirimkan adalah 200 yang berarti permintaan sedang diproses.

Protokol lapisan aplikasi (HTTP, dalam kasus ini) pada *client* (Bob) berkomunikasi dengan lapisan aplikasi pada *server* (Larry). Mereka melakukan itu dengan membuat dan mengirimkan *header* lapisan aplikasi satu sama lain. Bagaimanapun terlepas dari apa yang dilakukan oleh protokol lapisan aplikasi, semuanya menggunakan konsep yang sama untuk berkomunikasi dengan lapisanaplikasi pada komputer lain yang menggunakan *header* lapisan aplikasi.

2.2.4.2 TCP/IP Transport *Layer*

Lapisan *Transport* TCP/IP (TCP/IP *Transport Layer*) bertanggung jawab terhadap jaminan akan sampainya data ke tujuan. Lapisan ini terdiri dari dua protokol utama yaitu TCP (*Transmission Control Protocol*) dan UDP (*User Datagram Protocol*). Karena banyak protokol yang menginginkan jaminan akan sampainya data ketika data dikirim melewati jaringan, TCP menyediakan fitur *error-recovery* pada protokol aplikasi dengan menggunakan *acknowledgement*. Gambar 3.7. menjelaskan prinsip dasardarilogika *acknoledgement*.



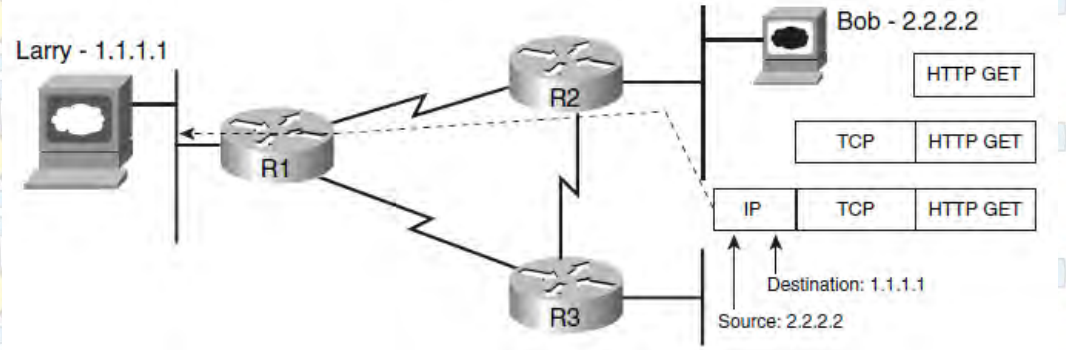
Gambar 1.8 Proses kerja TCP

Dari gambar 3.7. terlihat bahwa perangkat lunak HTTP meminta kepada TCP menyampaikan paket *request* dari HTTP. TCP mengirimkan data HTTP dari *client* (Bob) kepada *server* (Larry) dan data berhasil tiba dengan utuh. Protokol TCP yang bekerja pada *server* mengakui telah menerima data dan juga memberikan paket *request* HTTP kepada aplikasi *web server* dan hal sebaliknya juga terjadi pada respon *web server*. Pada kondisi normal seperti ini,kemampuan *error recovery* dari TCP belum akan terlihat dan baru akan terjadi jika data yang diterima terdapat kesalahan. TCP akan melakukan pengiriman data kembali untuk memastikan data dapat diterima dengan baik dan benar.

2.2.4.3 TCP/IP Internet *Layer*

Lapisan internet TCP/IP yang secara umum didefinisikan dengan *internet protocol* (IP) bekerja layaknya layanan pos. IP yang mendifinisikan alamat tiap komputer sehingga tiap komputer memiliki IP *address* yang berbeda layaknya layanan pos mendifinisikan alamat yang unik untuk pengalamatan tiap rumah, apartemen, atau kantor. IP menjelaskan proses *routing* sehingga perangkat *router* dapat memilih kemana harus mengirim paket data sehingga dapat sampai ke tempat yang benar. Seperti halnya layanan pos membuat infrastruktur pendukung untuk memungkinkan pengiriman surat seperti kantor pos, mesin penyortir, truk, pesawat, dan pegawai, lapisan internet juga menjelaskan secara mendetil bagaimana sebuah infrastruktur jaringan seharusnya dibangun sehingga jaringan

dapat mengirimkan data ke semua komputer dalam jaringan. Untuk memahami proses kerja dari lapisan internet dapat dilihat pada gambar 3.8. berikut.



Gambar 1.9 Layanan IP diberikan kepada TCP

Dari gambar diatas terlihat bahwa *header* IP mengandung informasi mengenai IP address sumber dan tujuan dengan IP tujuan (*destination*) 1.1.1.1 (Larry) dan IP sumber (*source*) 2.2.2.2 (Bob).

Bob mengirim paket menuju R2. R2 kemudian memeriksa IP *address* tujuan (1.1.1.1) dan membuat sebuah *routing decision* untuk mengirim paket menuju R1. Karena memiliki informasi secara jelas mengenai topologi jaringan, R2 mengetahui bahwa 1.1.1.1 (Larry) berada di sisi lain dari R1. Begitu juga dengan sebaliknya jika R1 menerima paket, R2 juga akan melakukan cara yang sama untuk mengirimkan paket menuju Larry. Jika *link* antara R2 dan R1 mengalami masalah, IP akan memberikan R2 kemampuan untuk mencari rute terbaik agar dapat melewati R3 menuju 1.1.1.1.

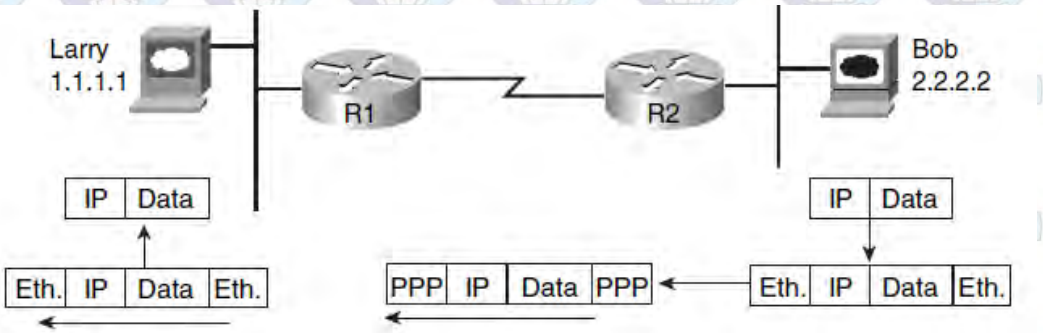
IP menefinisikan pengalamatan logis (*logical addresses*) yang disebut sebagai IP *addresses* yang memungkinkan setiap perangkat TCP/IP (disebut IP *host*) memiliki alamat agar dapat saling berkomunikasi.

2.2.4.4 TCP/IP Network Access *Layer*

Lapisan *network access* terdiri dari protokol dan perngkat keras (*hardware*) yang bertujuan menyampaikan data menyeberangi beberapa perangkat fisik jaringan. Misalnya, Ethernet adalah salah satu contoh protokol yang bekerja

pada lapisan *network access*. Ethernet terdiri dari pengkabelan, pengalamatan, dan protokol yang digunakan untuk membangun jaringan Ethernet.

Lapisan *network access* terdiri dari banyak protokol. Misalnya, lapisan *network access* terdiri dari berbagai variasi protokol Ethernet dan standar LAN lainnya. Lapisan ini juga terdiri dari standar WAN (*Wide Area Networks*) seperti Frame Relay dan PPP (*Point to Point Protocol*). Model jaringan yang cukup populer dapat dilihat pada gambar 3.9. Pada gambar tersebut dapat dilihat penerapan dari Ethernet dan PPP sebagai dua protokol lapisan *network access*.



Gambar 1.10 Penggunaan Ethernet dan PPP

Untuk mengirim paket menuju Larry, Bob mengirimkan paket menuju *router* (R2). Untuk melakukan itu, Bob menggunakan Ethernet untuk mendapatkan paket dari R2 dan proses tersebut mewajibkan Bobuntuk mengikuti aturan protokol Ethernet dengan menempatkan paket IP (*header* IP dan data) di antara *header* Ethernet dan *trailer* Ethernet.

Karena tujuan utama dari proses *routing* adalah menyampaikan paket IP menuju *host* tujuan, R2 tidak lagi membutuhkan *header* dan *trailer* Ethernet yang diterima dari Bob. Dengan demikian maka R2 akan menghilangkan *header* dan *trailer* Ethernet, meninggalkan paket IP yang asli. Untuk mengirim paket IP dari R2 menuju R1, R2 menempatkan *header* PPP di depan paket IP dan sebuah *trailer* PPP pada bagian akhir dan mengirimkan *frame* data melewati jaringan WAN menuju R1. Begitupula dengan sebaliknya, ketika paket diterima oleh R1, R1 akan menghilangkan *header* dan *trailer* PPP karena tugas dari PPP adalah mengirimkan paket IP menyebrangi jaringan WAN. R1 kemudian memutuskan

bahwa paket tersebut harus diteruskan melewati jaringan Ethernet menuju Larry. Untuk melakukan itu, R1 menambahkan *header* dan *trailer* Ethernet baru pada paket dan meneruskannya menuju Larry.

Kesimpulannya, lapisan TCP/IP *network access* termasuk protokol, standar pengkabelan, *header*, dan *trailer* mendefinisikan bagaimana mengirimkan data melewati berbagai variasi tipe dari jaringan fisik.

2.2.5 Teori Graf

Teori graf merupakan pokok bahasan yang sudah tua usianya namun memiliki banyak terapan saat ini. Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Representasi visual dari graf adalah dengan menyatakan objek dinyatakan sebagai noktaf, bulatan, atau titik, sedangkan hubungan antara objek dinyatakan dengan garis. Sebagai contoh, Gambar 2.11 adalah sebuah peta jaringan jalan raya yang menghubungkan sejumlah kota di Provinsi Jawa Tengah. Sesungguhnya peta tersebut adalah sebuah graf, yang dalam hal ini kota dinyatakan sebagai bulatan sedangkan jalan dinyatakan sebagai garis. Dengan diberikannya peta tersebut, kita dapat mengetahui apakah ada lintasan jalan antara dua buah kota. Selain itu, bila panjang jalan kereta api antara dua buah kota bertetangga diketahui, kita juga dapat menentukan rute perjalanan yang tersingkat dari kota A ke kota B.

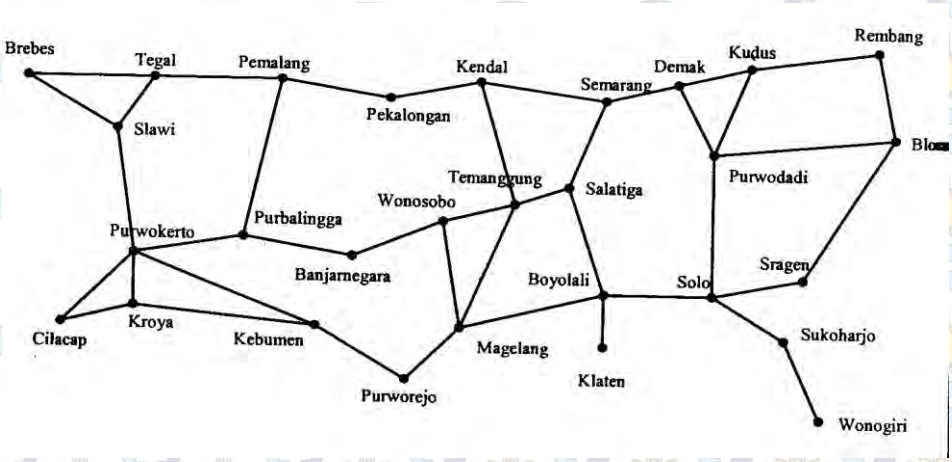
Cara merepresentasikan sebuah graf yang paling umum adalah dengan diagram. Tiap-tiap diagram memuat sekumpulan objek (titik) dengan garis-garis yang menghubungkan objek-objek tersebut. Garis bisa berarah ataupun tidak berarah. Garis yang berarah digunakan untuk menyatakan hubungan yang mementingkan urutan antar objek-objek. Urut-urutan objek akan mempunyai arti lain jika arah garis diubah. Garis yang tidak berarah digunakan untuk menyatakan hubungan antar objek-objek yang tidak mementingkan urutan. Dalam menggambarkan sebuah graf, bentuk sisi dapat berupa ruas garis/sisi lurus atau lengkung.

Setiap sisi berhubungan dengan satu atau dua titik. Titik-titik tersebut dinamakan titik ujung. sisi yang hanya berhubungan dengan satu titik ujung disebut Loop. Dua sisi berbeda yang menghubungkan titik yang sama disebut sisi paralel. Dua titik dikatakan berhubungan langsung (adjacent) jika ada sisi yang

menghubungkan keduanya. Titik yang tidak mempunyai sisi yang berhubungan dengannya disebut titik terasing (Isolating Point).

2.2.5.1 Sejarah Graf

Menurut catatan sejarah, masalah Jembatan Konisberg adalah masalah yang pertama kali menggunakan graf (tahun 1736). Di Kota Konisberg (sebelah timur negara bagian Prussia, Jerman), sekarang bernama kota Kaliningrad, terdapat Sungai Pregal yang mengalir mengitari Pulau Kneiphof lalu bercabang menjadi dua buah anaksungai.

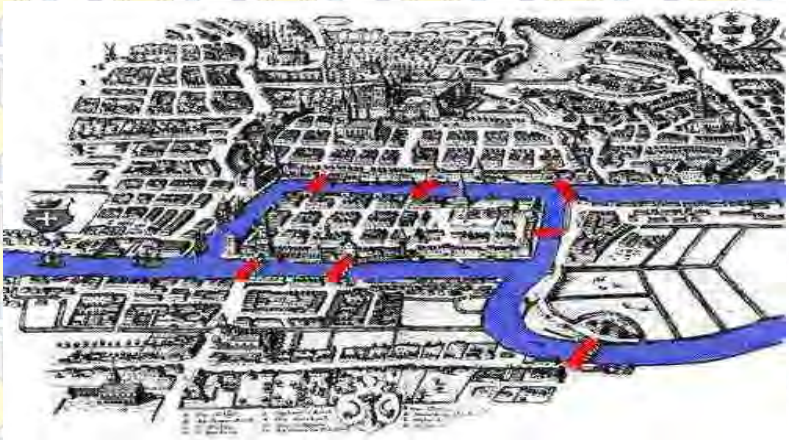


Gambar 1.11 Jaringan jalan raya di Provinsi Jawa Tengah

Ada tujuh buah jembatan yang menghubungkan daratan yang dibelah oleh sungai tersebut. Masalah Jembatan Konisberg adalah : apakah mungkin melalui ketujuh jembatan itu masing-masing tepat satu kali, dan kembali lagi ke tempat semula ? Sebagian penduduk kota sepakat bahwa memang tidak mungkin melalui setiap jembatan itu hanya sekali dan kembali lagi ke tempat asal mula keberangkatan, tetapi mereka tidak dapat menjelaskan mengapa demikian jawabannya, kecuali dengan cara coba-coba. Tahun 1736, seorang matematikawan Swiss, Leonhard Euler, adalah orang pertama yang berhasil menemukan jawaban masalah itu dengan pembuktian yang sederhana. Ia memodelkan masalah ini ke dalam Graf. Daratan (titik-titik yang dihubungkan oleh jembatan) dinyatakan sebagai titik (noktah) yang disebut simpul (*vertex*) dan jembatan dinayatakn

sebagai garis yang disebut sisi (*edge*). Setiap titik diberi label huruf *A*, *B*, *C*, dan D. Graf yang dibuat oleh Euler diperlihatkan pada Gambar 2.12.

Jawaban yang dikemukakan oleh Euler adalah : orang tidak mungkin melalui ketujuh jembatan itu masing-masing satu kali dan kembali lagi ke tempat asal keberangkatan jika derajat setiap simpul tidak seluruhnya genap. Yang dimaksud dengan derajat adalah banyaknya garis yang bersisihan dengan noktah. Sebagai contoh, simpul *C* memiliki derajat 3 karena ada tiga buah garis yang bersisian dengannya, simpul *B* dan *D* juga berderajat dua, sedangkan simpul *A* berderajat 5. Karena tidak semua simpul berderajat genap, maka tidak mungkin dilakukan perjalanan berupa sirkuit (yang dinamakan dengan sirkuit Euler) pada graf tersebut.



Gambar 1.12 Peta kota Konigsberg kuno dan jembatan bersejarahnya

2.2.5.2 Komponen Graf

Ada beberapa terminologi dari teori graf yang digunakan untuk menjelaskan apa yang dilihat ketika melihat suatu graf. Graf dapat dilihat dari komponen-komponen penyusunnya.

1. Titik (Verteks)

   Titik (Verteks) yang disimbolkan dengan v adalah himpunan titik yang terbatas dan tidak kosong. Jumlah titik pada graf dapat dinyatakan dengan n = |v|.

2. Sisi (Edge)

   Sisi (edge) yang disimbolkan dengan e adalah himpunan sisi yang menghubungkan sepasang titik.

3. Derajat (Degree)

Derajat (Degree) suatu titik yang disimbolkan dengan d(v) adalah jumlah sisi yang berada pada titik tersebut

4. Ukuran (Size)

Ukuran (Size) dari suatu graf adalah banyaknya titik yang dimiliki.

2.2.5.3 Keterhubungan

1. Jalan (Walk)

Misalkan G suatu graf dengan vi dan vj adalah 2 titik dalam G. Jalan (walk) dari vi ke vj adalah barisan titik dan sisi yang berhubungan secara bergantian, yang diawali dari titik vi dan diakhiri titik vj. Titik vi dan vj adalah titik awal dan akhir, sedangkan titik-titik yang yang berada di antara vi dan vj adalah titik-titk internal.

2. Jejak (Trail)

Jejak (trail) adalah jalan dengan sisi-sisi yang berbeda atau tanpa sisi berulang.

3. Lintasan (Path)

Lintasan (path) adalah jalan dengan titik dan sisi yang berbeda atau jejak dengan simpul yang berbeda.

4. Sirkuit

Sirkuit adalah jejak tertutup. Jejak tertutup adalah jejak dengan titik awal dan titik akhir sama.

5. Sirkuit Euler

Sirkuit Euler adalah sirkuit yang memuat semua sisi.

6. Jejak Euler

Jejak Euler adalah jejak yang memuat semua sisi.

7. Sikel (Cycle)

Sikel (Cycle) adalah sebuah jejak tertutup dengan titik awal dan semua titik internalnya berbeda.

8. Sikel Hamilton

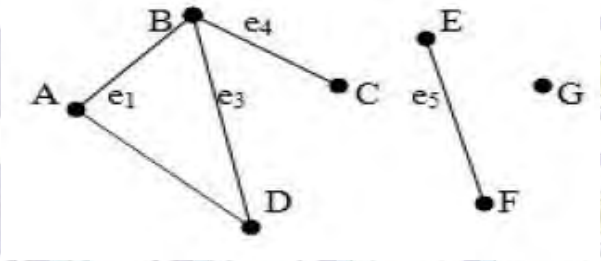
Sikel Hamilton adalah sikel untuk setiap titik di G yang dilalui tepat satu kali dan setiap sisi di G tidah harus dilalui.

Contoh 1.

Ada 7 desa (A, ..., G) yang akan dipasang pipa air, beberapa diantaranya dapat dihubungkan secara langsung. Hubungan-hubungan langsung yang dapat dilakukan adalah sebagai berikut : A dengan B dan D ; B dengan D ; C dengan B ; E dengan F. Buatlah graf yang menunjukkan keadaan sambungan pipa antar desa tersebut.

Penyelesaian:

Misalkan desa-desa dianggap sebagai titik-titik. Dua titik (desa) dihubungkan dengan garis jika dan hanya jika ada pipa yang menghubungkan langsung kedua kota tersebut. Dengan demikian, keadaan jalur pipa di 7 desa dapat dinyatakan dalam gambar di bawah ini.



Gambar 1.13 Graf dari contoh 1

Pada graf tersebut e1 berhubungan dengan titik A dan B (keduanya disebut titik ujung e1). Titik A dan B dikatakan berhubungan, sedangkan titik A dan G tidak berhubungan karena tidak ada sisi yang menghubungkannya secara langsung. Titik G adalah titik terasing karena tidak ada sisi yang berhubungan dengan G. Dalam interpretasinya, desa G merupakan desa yang terasing karena tidak dapat dipasangi pipa dari desa lain.

2.2.5.4 Jenis-jenis Graf

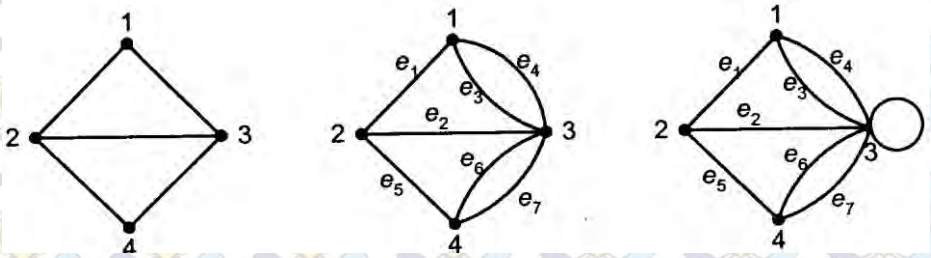
Graf dapat dikelompokkan menjadi beberapa kategori bergantung pada sudut pandang pengelompokannya. Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang, berdasarkanjumlah simpul, atau berdasarkan orientasi arah pada sisi.

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf,maka secara umum graf dapat digolongkan menjadi dua jenis, yaitu :

1. Graf sederhana (*simple graph*)

   Graf yang tidak mengandung gelang maupun sisi ganda dinamakan graf sederhana. Pada gambar 2.13(a) adalah contoh graf sederhana yang merepresentasikan jaringan komputer. Simpul menyatakan jaringan komputer, sedangkan sisi menyatakan saluran telepon untuk berkomunikasi. Saluran telepon dapat berorientasi pada dua arah.

   Pada graf sederhana, sisi adalah pasangan tak terurut (*unordered pairs*). Jadi, menuliskan sisi (u,v) sama saja dengan (v,u). Kita dapat mendefinisikan graf sederhana G = (V,E) terdiri dari himpunan tidak kosong simpul-simpul dan E adalah himpunan pasangan tak terurut yang berbeda yang disebut sisi.



Gambar 1.14 Tiga buah Graf terdiri dari graf sederhana, graf ganda, dan graf semu

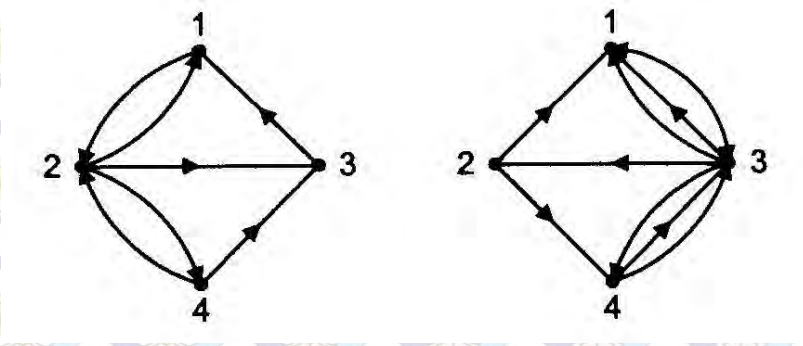
2. Graf tak sederhana (*unsimple graph*)

   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak sederhana. Ada dua macam macam graf tak sederhana, yaitu graf ganda (*multigraph*) dan graf semu (*pseudograph*). Graf ganda adalah graf yang mengandung sisi ganda. Sisi ganda yang menghubungkan sepasang simpul bisa lebih dari dua buah. Gambar kedua pada gambar 2.13 adalah graf ganda. Sisi ganda dapat diasosiasikan sebagai pasangan tak berurut yang sama. Kita dapat juga mendefinisikan graf ganda G = (V,E) terdiri dari himpunan kosong simpul-simpul dan E adalah himpunan ganda (*multiset*) yang mengandung sisi ganda. Pada jaringan telekomunikasi,sisi ganda

pada G_2 dapat diandaikan sebagai saluran telepon tambahan apabila beban komunikasi data antar komputer sangat padat.

Graf semu adalah graf yang mengandung gelang (*loop*). Gambar ketiga adalah graf semu (termasuk bila memiliki sisi ganda sekalipun). Sisi gelang pada G_3 dapat dianggap sebagai saluran telepon tambahan yang menghubungkan komputer dengan dirinya sendiri. Graf semu lebih umum daripada graf ganda, karena sisi pada graf semu dapat terhubung ke dirinya sendiri.

Jumlah simpul pada graf kita sebut sebagai kardinalitas graf, dan dinyatakan dengan $n = |V|$, dan jumlah sisi dinyatakan dengan $m = |E|$. Pada Gambar 2.11, G_1 mempunyai n = 4, dan m = 4, sedangkan G_2 mempunyai n = 3 dan m = 4.



Gambar 1.15 Graf berarah dan graf ganda berarah

Sisi pada graf dapat mempunyai orientasi arah. Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas dua jenis :

1. Graf tak-berarah (*undirected graph*)

   Graf yang sisinya tidak mempunyai orientasi arah disebut sebagai graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tak diperhatikan. Jadi, (u,v) = (v,u) adalah sisi yang sama. Pada jaringan telepon, sisi pada graf tak-berarah menyatakan bahwa saluran telepon dapat beroperasi pada dua arah.

2. Graf berarah (*directed graph*)

   Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Pada graf berarah, (u,v) dan (v,u) menyatakan dua busur yang

berbeda, dengan kata lain (u,v) ≠ (v,u). Untuk busur (u,v), simpul u dinamakan simpul asal (*initial vertex*) dan simpul terminal (*terminal vertex*). G_4 pada Gambar 2.12(a) adalah contoh graf berarah. Pada G_4 dapat dibayangkan sebagai saluran telepon tidak dapat beroperasi pada dua arah. Saluran hanya beroperasi pada arah yang ditunjukkan oleh anak panah. Jadi, sebagai contoh, saluran telepon (1,2) tidak sama dengan saluran telepon (2,1). Graf berarah sering dipakai untuk menggambarkan aliran proses, peta lalu lintas suatu kota (jalan searah atau dua arah), dan sebagainya. Pada graf berarah, gelang diperbolehkan, tetapi sisi ganda tidak.

2.2.5.5 Graf Berbobot

Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan dengan graf. Bobot pada menyatakan jarak antara dua buah kota, biaya perjalanan antara dua buah kota, waktu tempuh pesan (message) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya.

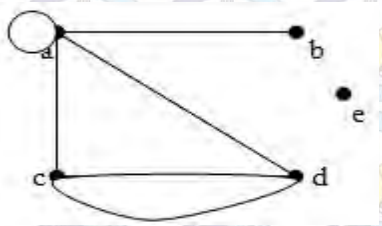
Istilah lain yang sering dikaitkan dengan graf berbobot adalah graf berlabel. Namun graf berlabel sesungguhnya lebih luas definisinya. Label tidak hanya diberikan pada sisi, tetapi juga pada simpul. Simpul diberi label berupa bilangan tak-negatif, sedangkan simpul diberi label berupa data lain. Misalnya pada graf yang memodelkan kota-kota, simpul diberi nama kota-kota, sedangkan label pada sisi menyatakan jarak antar kota-kota.

2.2.5.6 Matriks Ketetanggaan

Matriks dapat digunakan untuk menyatakan suatu graf. Hal ini sangat membantu untuk membuat program komputer yang berhubungan dengan graf. Dapat menyatakan graf sebagai suatu matriks, maka perhitungan-perhitungan yang diperlukan dapat dilakukan dengan mudah.

Matriks ketetanggaan atau matriks berhubungan langsung digunakan untuk menyatakan graf dengan cara menyatakannya dalam jumlah garis yang menghubungkan titik-titiknya. Jumlah baris (dan kolom) matriks ketetanggaan sama dengan jumlah titik dalam graf.

Misalkan G adalah sebuah graf dengan n titik. Matriks ketetanggaan dari graf G adalah matriks bujur sangkar (persegi) berordo n, X(G) = x(ij), dengan elemen x(ij) menyatakan banyaknya sisi yang menghubungkan titik ke-i ke titik ke-j. Dengan definisi ini memungkinkan untuk menyatakan sebuah graf yang memiliki sisi paralel atau loop dengan matriks ketetanggaan.



Gambar 1.16 Graf yang memiliki sisi parallel dan loop

$$X(H) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Gambar 1.17 Sebuah matriks ketetanggan

Matriks ketetanggaan juga digunakan untuk menyatakan graf berbobot, yaitu elemen-elemenya menyatakan bobot garis.

Diketahui G graf berbobot dengan setiap sisi dengan suatu bilangan riil tak negatif. Matriks yang bersesuaian dengan graf berbobot G adalah matriks ketetanggan atau matriks keterhubungan X(G) = x(ij) dengan xij = bobot garis yang menghubungkan titik vi dengan titik vj. Jika titik vi tidak berhubungan langsung dengan titik vj maka xij = ∞, dan xij = 0, jika i = j.

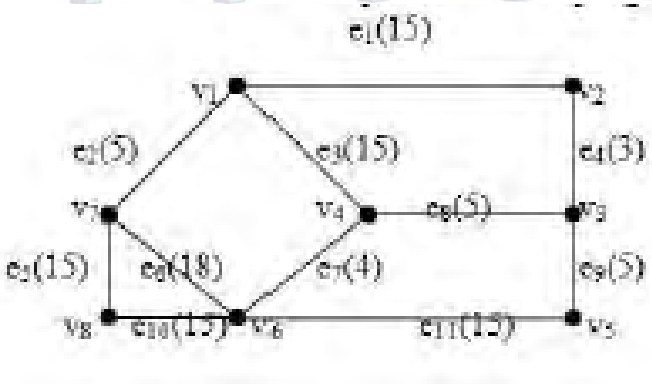
Sebagai contoh, dalam suatu propinsi, ada 8 kota (v1, v2, ..., v8) yang akan dihubungkan dengan jaringan-jaringan listrik. Biaya pemasangan jaringan listrik yang akan dibuat antar 2 kota adalah sebagai berikut.

Tabel 1.2 Contoh kasus graf

Garis	Desa yang dihubungkan	Biaya per satuan
e_4	$V_2 - V_3$	3

e_7	$V_4 - V_6$	4
e_2	$V_1 - V_7$	5
e_8	$V_3 - V_4$	5
e_9	$V_3 - V_5$	5
e_1	$V_1 - V_2$	15
e_3	$V_1 - V_4$	15
e_{10}	$V_6 - V_8$	15
e_5	V_7 - V_8	15
e_{11}	$V_5 - V_6$	15
e_6	$V_6 - V_7$	18

Maka penyelesaiannya, graf berbobot untuk menyatakan jaringan pipa di 8 desa digambarkan pada gambar di bawah ini. Angka dalam kurung menyatakan bobot garis yang bersangkutan. Bobot tersebut menyatakan biaya pemasangan jaringan listrik.



Gambar 1.18 Graf berbobot

Matriks keterhubungan untuk menyatakan graf berbobot pada gambar di atas adalah matriks X(G) = x(ij) dengan,

xij = bobot garis yang menghubungkan titik vi dengan titik vj,

xij = ∞, Jika titik vi tidak berhubungan langsung dengan titik vj, dan

xij = 0, Jika i = j.

$$X(G) = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{array}{c} \begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \end{array} \\ \begin{bmatrix} 0 & 15 & \infty & 15 & \infty & \infty & 5 & \infty \\ 15 & 0 & 3 & \infty & \infty & \infty & \infty & \infty \\ \infty & 3 & 0 & 5 & 5 & \infty & \infty & \infty \\ 15 & \infty & 5 & 0 & \infty & 4 & \infty & \infty \\ \infty & \infty & 5 & \infty & 0 & 15 & \infty & \infty \\ \infty & \infty & \infty & 4 & 15 & 0 & 18 & 15 \\ 5 & \infty & \infty & \infty & \infty & 18 & 0 & 15 \\ \infty & \infty & \infty & \infty & \infty & 15 & 15 & 0 \end{bmatrix} \end{array}$$

Gambar 1.19 Matriks ketetanggan yang dibentuk

Dalam program komputer, sel dengan harga ∞ diisi dengan suatu bilangan yang harganya jauh lebih besar dibandingkan dengan harga elemen-elemen yang bukan ∞.

2.2.6 Model Rute Terpendek

Model rute terpendek adalah salah satu model jaringan yang dapat digunakan untuk menentukan jarak terpendek dari berbagai alternatif rute yang tersedia atau mencoba untuk memecahkan masalah pemilihan jaringan paling efisien yang akan menghubungkan satu titik ke titik yang lain. Suatu lintasan antara dua buah titik adalah serangkaian garis yang berbeda yang menghubungkan titik-titik tersebut. Untuk setiap dua titik dapat terjadi beberapa lintasan, maupun lintasan dengan jarak terpendek atau bobot minimum. Bobot minimum dapat berupa jarak, waktu tempuh atau ongkos transportasi dari satu titik ke titik yang lainnya yang berbentuk lintasan tertentu. Rute terpendek yang dicari adalah lintasan dari sumber ke tujuan yang memecahkan persoalan jarak total minimum. Faktor-faktor yang mempengaruhi pemilihan lintasan diantaranya adalah waktu tempuh, jarak, ongkos, kemacetan, dan antrian. Terdapat beberapa macam persoalan lintasan terpendek sebagai berikut.

1. Lintasan terpendek antara dua buah titik tertentu.
2. Lintasan terpendek antara semua pasangan titik. Dapat diselesaikan dengan menggunakan algoritma Floyd-Warshall.

3. Lintasan terpendek dari titik tertentu ke semua titik yang lain. Dapat diselesaikan misalnya dengan menggunakan algoritma Dijkstra atau algoritma Bellman-Ford.
4. Lintasan terpendek antara dua buah titik yang melalui beberapa titik tertentu.

2.2.7 Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger Dijkstra pada tahun 1959, adalah algoritma pencarian graf yang memecahkan masalah jalur terpendek yang bersumber dari satu simpul untuk sebuah graf dengan bobot simpul tidak boleh negatif [6]. Analisis dilakukan dengan cara memeriksa simpul dengan bobot terkecil dan memasukkannya ke dalam himpunan solusi dengan awal pencarian simpul asal membutuhkan pengetahuan tentang semua jalur dan bobotnya, sehingga dibutuhkan pertukaran informasi dengan semua simpul. Algoritma dijkstra memiliki sifat yang sederhana dan lempeng (straightforward), sesuai dengan prinsip kerja greedy. Elemen-elemen penyusun algoritma *greedy* adalah:

1. Himpunan kandidat, C

   Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam graf, himpunan kandidat ini adalah himpunan simpul pada graf tersebut.
2. Himpunan solusi, S

   Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah upabagian dari himpunan kandidat.
3. Fungsi seleksi

   Fungsi seleksi adalah fungsi yang akan memilih setiap kandidat yang yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.
4. Fungsi kelayakan

   Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar *constraint* atau tidak. Apabila kandidat

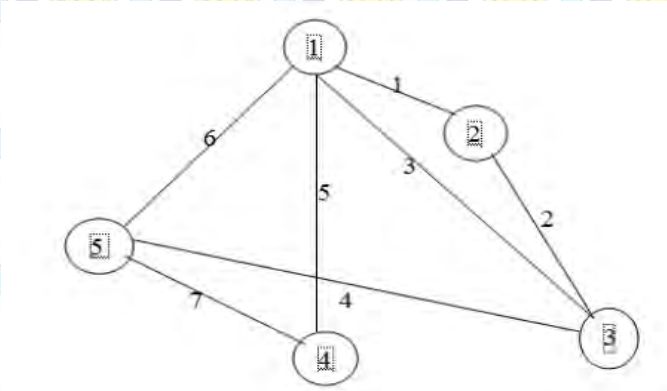
melanggar constraint maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

5. Fungsi objektif

   Fungsi objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi.

Gambar 3.10. di bawah ini diberikan contoh sebuah graf tak berarah yang terdiri dari 5 buah titik dan 7 buah jalur yang menghubungkan antar dua buah titik. Algoritma djikstra digunakan untuk mencari jarak terpendek dari sebuah titik ke titik lainnya pada graf tak berarah tersebut.

Berdasarkan contoh graf tak berarah di atas ditentukan titik awal pencarian adalah titik 1 dengan tujuan yaitu titik 4 dan akan dicari jarak terpendek yang dapat ditempuh dari titik 1 untuk menuju titik 4. Berikut ini tabel penjelasan graf menggunakan algoritma dijkstra:



Gambar 1.20 Contoh kasus graf tak berarah

Tabel 1.3 Penjelasan graf menggunakan algoritma dijkstra

Jalur	**Initial Jalur**					**Titik**	**I(i,j)**				
	1	**2**	**3**	**4**	**5**						
	0	0	0	0	0		**1**	**2**	**3**	**4**	**5**
1	1	0	0	0	0	**1**	∞	∞	∞	∞	∞

1-2	1	1	0	0	0	**2**	1	∞	∞	∞	∞
2-3	0	1	1	0	0	**3**	3	2	∞	∞	∞
3-5	0	0	1	0	1	**4**	5	∞	∞	∞	7
5-4	0	0	0	1	1	**5**	6	∞	4	∞	∞

Penyelesaian algoritma dijkstra jalur titik 1 ke titik 4 telah diselesaikan seperti pada penjelasan dan tabel di atas menurut perhitungan penelusuran graf sesuai langkah prosedural algoritma dijkstra. Pada baris pertama semua *successor* di set 0 artinya untuk memberi nilai pada sumber titik rute yang akan dijadikan rute dan ketidakterbatasaan untuk semua titik lain, yang menyatakan fakta bahwa tidak diketahui lintasan manapun.

Untuk selanjutnya karena titik 1 sebagai sumber lintasan maka sudah pasti terpilih. Sehingga status set 0 berubah menjadi 1. Titik 1 akan cek titik yang bertetanggaan langsung yaitu titik 2, 3, 4 dan 5. Dari situ dijkstra akan memilih yang mempunyai bobot terendah untuk menuju titik selanjutnya. Terpilih titik 2 dengan bobot 1, set status 0 berubah menjadi 1 dan seterusnya. Maka dari pencarian jarak terpendek di atas, didapat lintasan yang terpendek berdasarkan pencarian dijkstra dari titik 1 ke 4 adalah melalui titik 1 langsung titik 4 dengan bobot lintasan 5.

2.2.8 Modified Dijkstra Shortest Path Algorithm (MDSP)

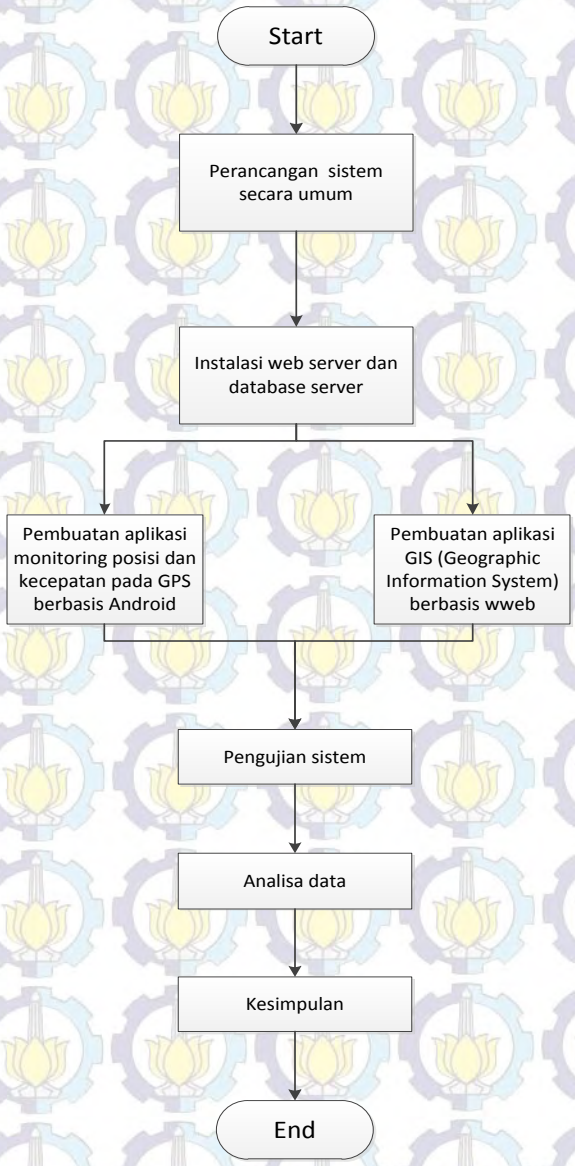
MDSP merupakan pengembangan atau modifikasi dari Algoritma Dijkstra yang digunakan untuk menentukan rute menuju sebuah node dengan menggunakan lebih dari satu parameter[17].

Pada pengembangan sebuah sistem informasi geografis (GIS) yang digunakan untuk memecahkan masalah pencarian jalur terbaik pada sebuah lingkungan jalan perkotaan, penggunaan parameter jarak sebagai bobot utama pada Algoritma Dijkstra tidaklah memadai sehingga dilakukan penambahan parameter seperti waktu atau faktor kemacetan sehingga hasil dari proses *routing* yang dilakukan lebih valid.

BAB III
METODE PENELITIAN

3.1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan yang meiliki fungsi dan target masing-masing. Tahapan dari penelitian yang dilakukan, dapat dilihat pada Gambar 3.1.



Gambar 1.1 Diagram alir penelitian

Dari diagram alir pada Gambar 3.1, maka tahapan – tahapan yang dilalui dalam penelitian ini adalah sebagai berikut:

3.1.1 Perancanaan sistem secara umum

Pada tahapan ini dilakukan perencanaan mengenai komponen apa saja yang diperlukan dalam pembuatan sistem ini seperti tipe perangkat GPS *receiver* serta sistem operasi yang akan digunakan pada unit *server* . Selain itu diperhitungkan pula mengenai parameter apa saja yang menentukan kinerja dari sistem yang akan dibangun. Desain dari sistem yang dibangun dapat dilihat pada Gambar 3.2.

3.1.2 Instalasi web server dan database server

Pada tahapan ini dilakukan proses instalasi terhadap server yang akan berfungsi menampilkan aplikasi peta berbasis web serta menyiapkan server yang berfungsi sebagai database bagi sistem yang dirancang. Aplikasi yang akan dipasang pada server adalah Apache untuk menjalankan aplikasi http dan MySql untuk menjalankan aplikasi *database*. *Server* yang akan digunakan akan memiliki *ip public* sehingga dapat diakses secara *online*.

3.1.3 Pembuatan aplikasi monitoring pada pada perangkat GPS

Pada tahap ini, dibangun aplikasi yang mampu melakukan *data acuisition* terhadap data posisi dan kecepatan dari tiap – tiap kendaraan. Selain itu aplikasi ini akan mampu mengirimkan data yang telah didapatkan menuju server menggunakan melalui jaringan GSM dan data akan dikirimkan menuju *server* dalam bentuk TCP *packet*. Aplikasi ini akan di *install* pada perangkat GPS berupa *smartphone* dengan sistem operasi Android dengan menggunakan bahasa pemrograman Java. Sebuah *smartphone* yang telah mengaplikasikan sistem GPS akan mampu melacak posisi dengan baik dan dapat diterapkan sebagai perangkat *tracking* pada sistem transportasi umum [10].

3.1.4 Pembuatan aplikasi GIS berbasis web.

Pada tahapan ini dibangun sebuah aplikasi GIS (*Geographic Information System*) yang berfungsi menampilkan secara mendetail setiap posisi dan kecepatan dari kendaraan yang dipantau melalui layanan berbasis *web* serta memberikan akses bagi calon pelanggan yang ingin melakukan pemesanan taksi. Aplikasi ini akan dijalankan pada unit *server* yang telah memiliki *ip public* sehingga dapat diakses dimana saja dengan menggunakan perangkat *mobile*. Seorang pelanggan yang akan melakukan pemesanan angkutan atau taksi dapat mengakses sistem ini secara *online*. Ketika pelanggan telah memasukkan posisinya saat ini, maka aplikasi ini akan langsung mencari taksi dengan posisi terdekat dari penumpang menggunakan Algoritma Dijkstra.

3.1.5 Pengujian Sistem

Setelah sistem selesai dibangun, tahap selanjutnya adalah dengan melakukan pengujian dari sistem yang dibangun ini. Pengujian ini dilakukan untuk mengetahui keakuratan dari kecepatan dan posisi kendaraan yang berhasil dilacak, besarnya *delay* yang terjadi ketika pengiriman data dilakukan dari kendaraan menuju ke server serta mengetahui waktu yang diperlukan oleh sistem untuk melakukan *reponse* terhadap setiap permintaan oleh pelanggan.

Selain pengujian untuk mengetahui apakah sistem mampu melakukan pencarian secara cepat terhadap taksi dengan kriteria terbaik, dilakukan juga simulasi dengan menambahkan jumlah node secara bertahap mulai dari 10 sampai dengan 50 untuk mengetahui seberapa cepat sistem mampu mencari node dengan kriteria terbaik dengan jumlah yang banyak.

Faktor terakhir yang akan diuji adalah seberapa mampu sistem yang dibangun ini mampu mencari node terbaik pada kondisi tingkat kepadatan jalan yang beragam. Pada skenario pengujian ini, digunakan Algoritma MDSP (*Modified Dijkstra Shortest Path*) yang dapat mengakomodasi penggunaan lebih dari satu parameter dalam memecahkan masalah pencarian jalur atau node terbaik. Pada simulasi ini akan digunakan tiga kondisi yaitu :

Tabel 1.1 Skala tingkat kepadatan pada simulasi pengujian sistem

Tingkat Kepadatan	Nilai	Warna node
Lancar	1	Hitam
Padat bergerak	3	Kuning
Macet total	5	Merah

Nilai tingkat kepadatan akan diperkalikan dengan parameter jarak sesuai dengan aturan yang terdapat pada Algoritma MDSP dengan rumus sebagai berikut :

W = d x s

dimana :

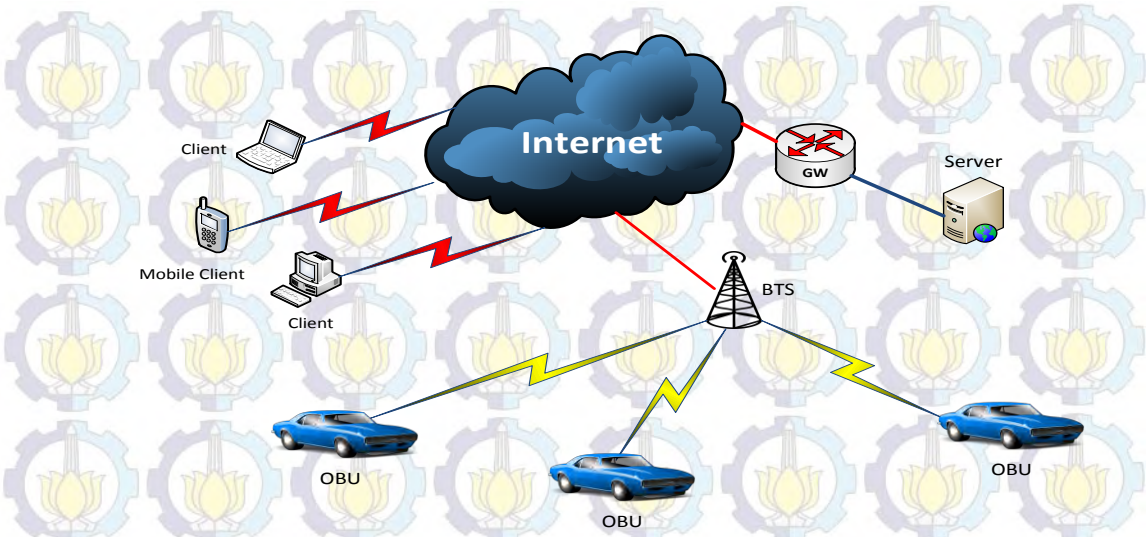
W = Bobot jalur

d = Jarak antar node

s = Tingkat kepadatan

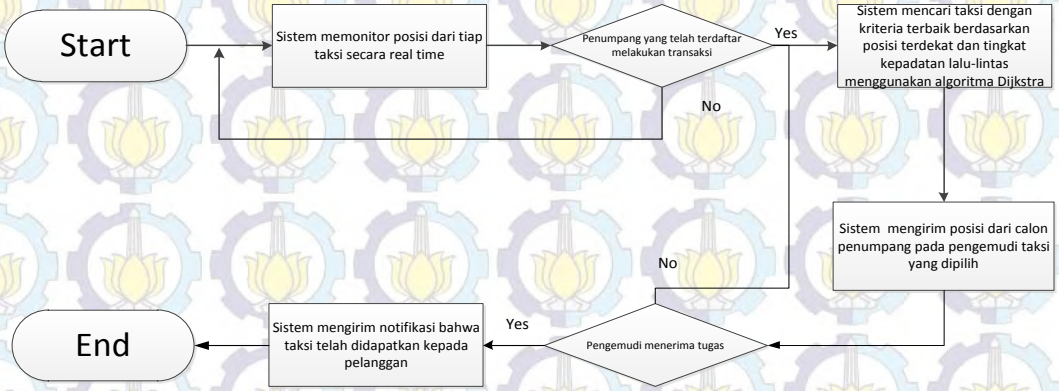
3.1.6 Analisis dan kesimpulan

Tahapan akhir dari penelitian ini adalah analisa data dan penarikan kesimpulan. Setelah data dari proses pengujian sistem berhasil didapatkan, maka akan dilakukan analisa terhadap berbagai faktor yang mempengaruhi performa atau kinerja sistem secara keseluruhan. Gambaran Umum Sistem Sistem yang ini terdiri dari perangkat GPS *receiver* berbasis *smartphone* android yang ditempatkan pada tiap kendaraan serta unit *server* yang berfungsi menerima informasi posisi dari tiap GPS *receiver* dan menampilkannya melalui layanan berbasis *web*. Sistem ini bekerja dengan melakukan *monitoring* secara *real time* terhadap posisi kendaraan, dan tiap pengendara serta tingkat kepadatan arus lalu lintas. Sistem ini akan secara otomatis menghubungi pengendara taksi jika pelanggan telah sepakat dengan transaksi yang terjadi.



Gambar 1.2 Rancangan arsitektur sistem

Sistem ini akan menggunakan Algoritma Dijkstra dalam proses *routing* dengan tujuan mencari taksi yang paling baik pelanggan. Parameter yang digunakan sebagai bobot atau acuan dalam menentukan taksi yang paling cocok bagi penumpang yang melakukan pememsanan adalah taksi dengan jarak terdekat serta tingkat kepadatan lalu – lintas yang terjadi antara pelanggan dengan tiap – tiap taksi yang ada. Proses yang terjadi pada sistem ini dapat dilihat pada diagram proses bisnis pada gambar 3.3.



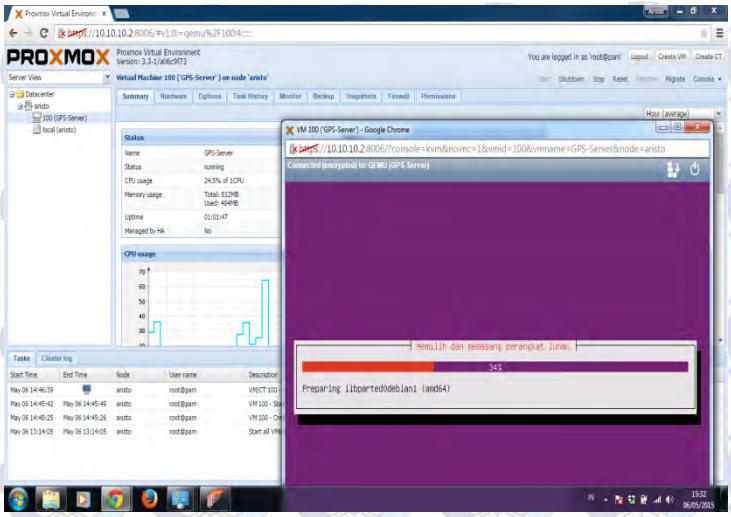
Gambar 1.3 Diagram proses bisnis pada sistem

3.2 Pembuatan Sistem

Agar sistem ini dapat berjalan dengan baik, diperlukan instalasi dan konfigurasi dari berbagai perangkat yang akan digunakan baik *hardware* maupun *software*. Tahapan pembuatan dari sistem ini dapat dilihat sebagai berikut.

3.2.1 Instalasi unit server

Server yang digunakan pada sistem ini menggunakan sistem operasi Linux Ubuntu 12.04 yang telah ditingkatkan menjadi Ubuntu 14.04. Server ini dijalankan pada sebuah mesin virtual yaitu Proxmox 1.9 dengan tujuan memudahkan proses pembuatan *backup server* jika salah satu *server* mengalami masalah. Server ini mendapatkan *ip address public* sehingga dapat diakses dimana saja.

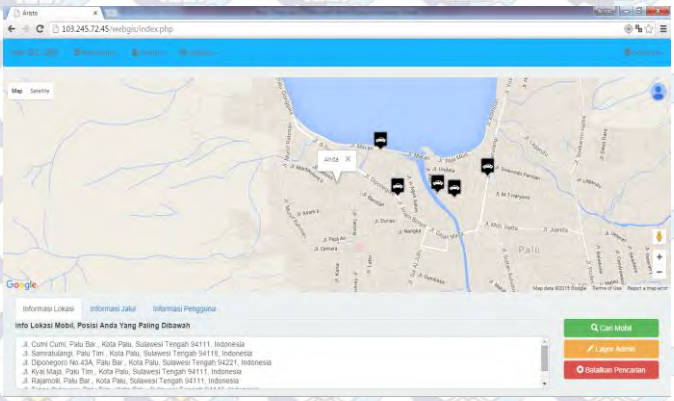


Gambar 1.4 Proses instalasi *server*

3.2.2 Pembuatan aplikasi webgis untuk tracking dan routing

Aplikasi webgis yang digunakan pada sistem ini dibangun dengan menggunakan bahasa pemrograman PHP dan Javascript serta menggunakan database MySQL. Aplikasi ini dapat diakses dengan mudah oleh pelanggan maupun operator taksi menggunakan laptop atau *smartphone*. Melalui aplikasi ini sistem mampu melakukan pelacakan (*tracking*) terhadap posisi tiap-tiap taksi dan menampilkannya secara *online*. Sistem mampu mengetahui lokasi tiap-tiap taksi berdasarkan pada data *latitude* dan *longitude* yang dikirimkan oleh aplikasi GPS *tracker* yang ditanamkan pada mmasing-masing *smartphone* pengemudi taksi. Dengan demikian, maka sistem ini juga akan mampu mengetahui taksi dengan posisi paling dekat dari pelanggan.

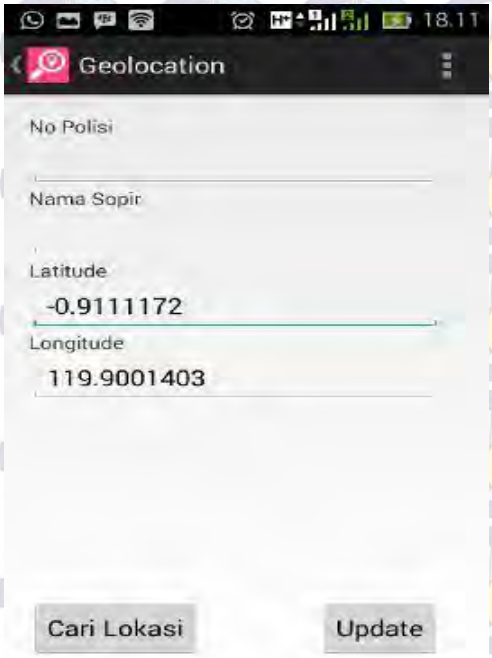
Gambar 1.5 Tampilan aplikasi berbasis *mobile*.



Gambar 1.6 Tampilan aplikasi melalui PC.

3.2.3 Pembuatan aplikasi GPS Tracker dan Push E-mail

Pada *smartphone* Android ditanamkan aplikasi *tracker* yang berfungsi melakukan pengiriman koordinat lokasi pengendara secara *real time* menuju server berupa data *latitude* dan *longitude* dari *smartphone* tiap pengendara. Selain aplikasi *tracker*, aplikasi lain yang ditanamkan dalam *smartphone* pengendara adalah aplikasi layanan komunikasi berbasis push e-mail yang berfungsi mengirimkan pesan yang berisi informasi lokasi dari pelanggan yang melakukan pemesanan taksi. Setiap pengendara taksi dapat menerima atau menolak *order* yang dikirimkan melalui aplikasi ini.



Gambar 1.7 Aplikasi GPS Tracker

Gambar 1.8 Aplikasi Push E-mail

3.3 Skenario Pengujian

Pengujian ini dilakukan pada wilayah Kota Palu, Provinsi Sulawesi Tengah. Data yang didapatkan dari pengujian yang dilakukan akan dibandingkan dengan aplikasi sejenis yang pernah dibuat sebelumnya untuk mengetahui keunggulan dari sistem yang dibuat.

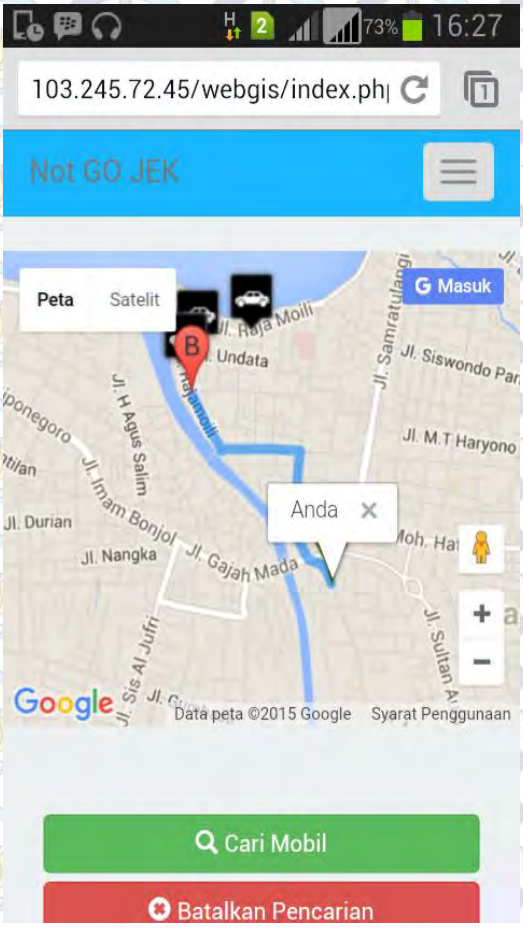
Parameter yang akan digunakan sebagai acuan dalam menentukan kinerja dari sistem ini adalah kemampuan sistem untuk mencari dan menemukan taksi dengan posisi yang paling dekat dari pelanggan diantara semua taksi yang ada dan tersebar di berbagai lokasi.

Parameter lain yang digunakan sebagai acuan adalah kecepatan layanan dari sistem yang diukur mulai dari ketika pelanggan melakukan pemesanan (reservasi), ketika pengemudi taksi mendapatkan notifikasi dari sistem, hingga pelanggan menerima notifikasi dari sistem yang dibangun. Pengujian dari waktu pencarian akan dilakukan dengan menambahkan jumlah node yang ada secara bertahap untuk mengetahui seberapa besar pengaruh dari jumlah armada yang tersedia di suatu perusahaan taksi terhadap kecepatan layanan dari sistem yang dibangun.

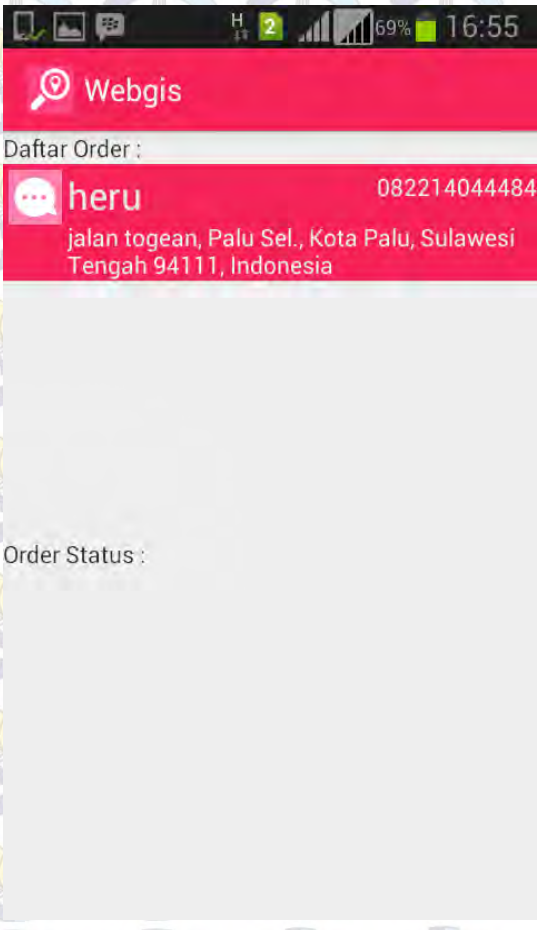
BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan ditampilkan implementasi dan hasil dari pengukuran yang dilakukan pada sistem yang telah dibangun berdasarkan dari skenario pengujian pada bab empat.

1.1 Tampilan Aplikasi



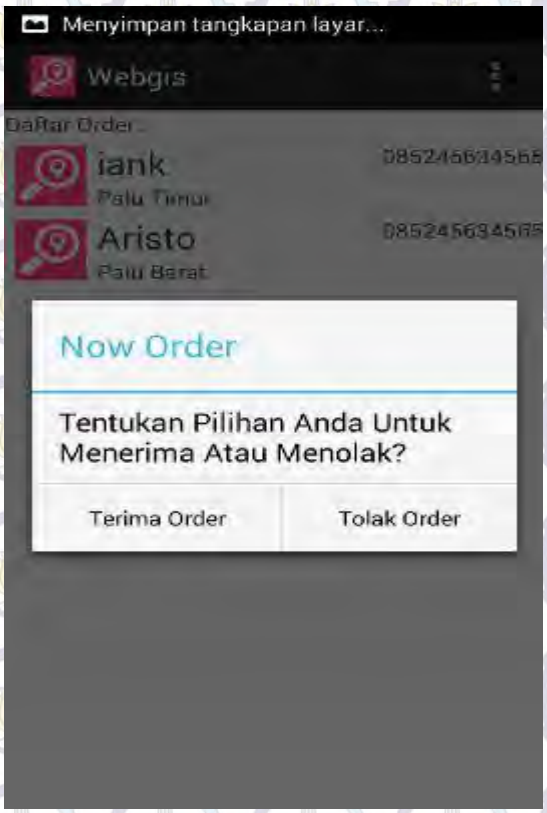
Gambar 1.1 Sistem mencari taksi terdekat



Gambar 1.2 Notifikasi pada pengendara taksi

Interface dari sistem yang telah dibangun dapat diakses melalui layanan berbasis *web*. Ketika sistem telah diakses dan penumpang telah setuju dengan

transaksi yang telah dilakukan maka sistem akan melakukan pencarian dengan posisi paling dekat dari pelanggan seperti yang dapat dilihat pada Gambar 4.1.



Sistem akan mengirimkan notifikasi berupa alamat dari pelanggan kepada pengemudi taksi melalui layanan berbasis push e-mail seperti yang dapat dilihat pada Gambar 4.2. Melalui layanan ini pengemudi taksi dapat memilih untuk menerima atau menolak pemesanan. Jika pengemudi menolak, maka sistem dapat melakukan pencarian kembali taksi terdekat. Setiap tanggapan yang dilakukan oleh pengemudi taksi baik menerima atau menolak order, pelanggan akan menerima tanggapan dari sistem melalui layanan SMS (*Shor Message Service*).

Gambar 1.3 Notifikasi pesanan ditolak

Gambar 1.4 Notifikasi pesanan diterima

1.2 Proses Pencarian Taksi Terdekat

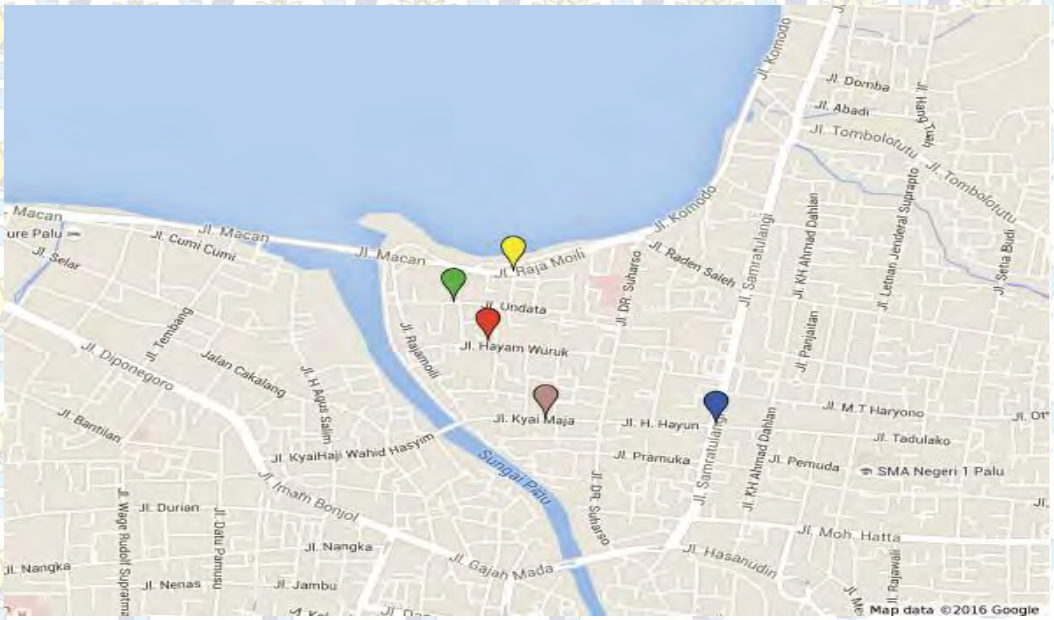
Proses yang berjalan pada sistem yang dibangun ini dapat dipahami dengan mengikuti skenario pengujian berikut. Pengujian dilakukan dengan melibatkan tujuh orang yang terdiri dari enam orang yang bertindak sebagai pengendara taksi dan satu sebagai pelanggan. Posisi dari pengendara disebar ke berbagai sudut kota untuk menyesuaikan dengan kondisi yang terjadi sehari-hari. Posisi dari tiap pengemudi dapat dilihat pada Tabel 4.1.

Data posisi terbaru dari penumpang dan pengendara taksi yang berupa data *latitude* dan *longitude* disimpan di dalam database pada server yang digunakan. Di dalam database akan ditampung posisi seluruh taksi, jarak antara tiap taksi satu sama lain dan jarak antara tiap taksi dengan pelanggan yang melakukan transaksi.

Tabel 1.1 Pengendara dan posisi masing-masing

No	Inisial	Alamat
1	**M1**	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia
2	**M2**	Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia
3	**M3**	Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia
4	**M4**	Jl. Kyai Maja, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia
5	**M5**	Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia
6	**M6**	Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia

Pengujian dilakukan di Kota Palu, Provinsi Sulawesi Tengah. Posisi dari pelanggan dan pengendara dapat dilihat pada Gambar 4.5 berikut.

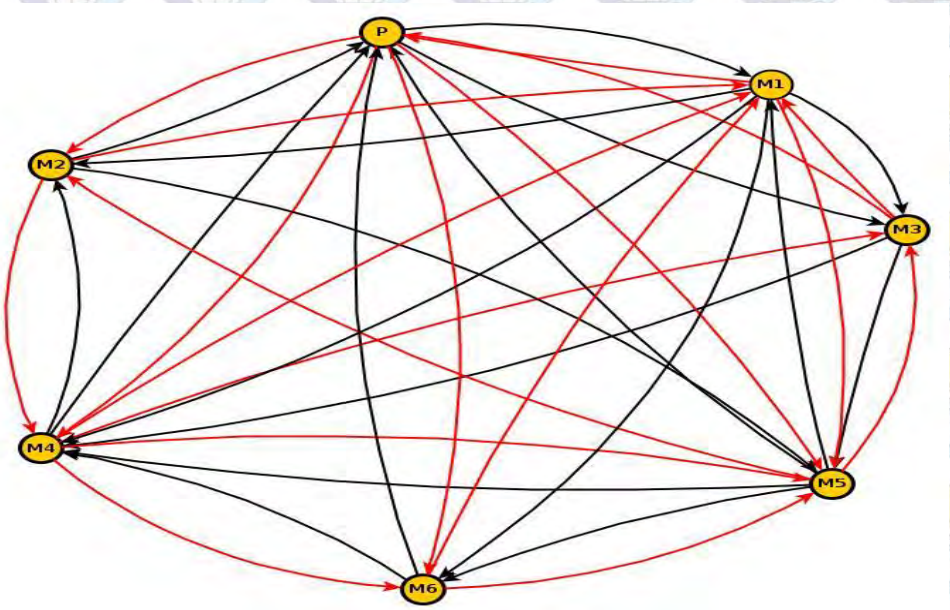


Gambar 1.5 Posisi dari tiap pengendara pada peta (warna hijau adalah posisi pelanggan)

Berdasarkan data posisi dan jarak antara penumpang dan tiap taksi yang terkumpul pada server, maka dapat dibangun sebuah model graf yang menggambarkan hubungan antara pelanggan dengan masing-masing taksi serta hubungan antara tiap-tiap taksi seperti yang dapat dilihat pada Gambar 4.6. Model

graf yang digunakan dalam sistem ini adalah graf berarah yang disesuaikan dengan kondisi lalu lintas jalan di tempat pengujian dilakukan.

Ketika pelanggan memilih *button* "cari mobil" pada layar aplikasi, maka Algoritma Dijkstra yang merupakan algoritma greedy pada sistem akan aktif mencari node apa saja yang terhubung dengan node P (penumpang) berdasarkan model graf diatas serta menghitung bobot dari tiap jalur yang ada dan menempatkannya dalam sebuah matriks adjacency dan hasilnya telah dirangkum pada tabel 4.2.



Gambar 1.6 Ilustrasi hubungan antara penumpang dengan tiap pengendara taksi dan antara pengendara taksi lainnya dalam graf

Tabel 1.2 Penumpang dan pengemudi-pengemudi taksi yang tersedia

Jalur	Pengguna	Start	Finish	Jarak
1	P1	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia (M1)	Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	1590
2			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	5267
3			Jl. Kyai Maja, Palu Tim., Kota Palu,	1473

			Sulawesi Tengah 94111, Indonesia	
4			Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	2290
5			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	5267
6			Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	1926
7		Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia (M2)	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia	1770
8			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6672
9			Jl. Kyai Maja, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	629
10			Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	622
11			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6672
12			Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	442
13		Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia (M3)	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia	5267
14			Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	6446
15			Jl. Kyai Maja, Palu Tim., Kota Palu,	6686

			Sulawesi Tengah 94111, Indonesia	
16			Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	7145
17			Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	6781
18		Jl. Kyai Maja, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia (M4)	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia	1971
19			Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	591
20			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6874
21			Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	393
22			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6874
23			Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	989
24		Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia (M5)	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia	2088
25			Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	494
26			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6990
27			Jl. Kyai Maja, Palu Tim., Kota Palu,	947

			Sulawesi Tengah 94111, Indonesia	
28			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6990
29			Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	604
30		Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia (M6)	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia	5267
31			Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	6446
32			Jl. Kyai Maja, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	6686
33			Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	7145
34			Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia	6781
35		Jl. Undata, Palu Bar., Kota Palu, Sulawesi Tengah 94111, Indonesia (P1)	Jl. Samratulangi, Palu Tim., Kota Palu, Sulawesi Tengah 94118, Indonesia	1954
36			Jl. Hayam Wuruk, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	442
37			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6857
38			Jl. Kyai Maja, Palu Tim., Kota Palu, Sulawesi Tengah 94111, Indonesia	1065
39			Jl. Rajamoili, Palu Bar., Kota Palu, Sulawesi	604

			Tengah 94111, Indonesia	
40			Jl. Trans Sulawesi, Palu Tim., Kota Palu, Sulawesi Tengah 94148, Indonesia	6857

Berdasarkan Tabel 4.2 yang berisi posisi dari pelanggan dan tiap-tiap taksi serta jarak dari tiap-tiap item, maka jarak dari tiap taksi menuju pelanggan dapat dirangkum ke dalam Tabel 4.3 berikut.

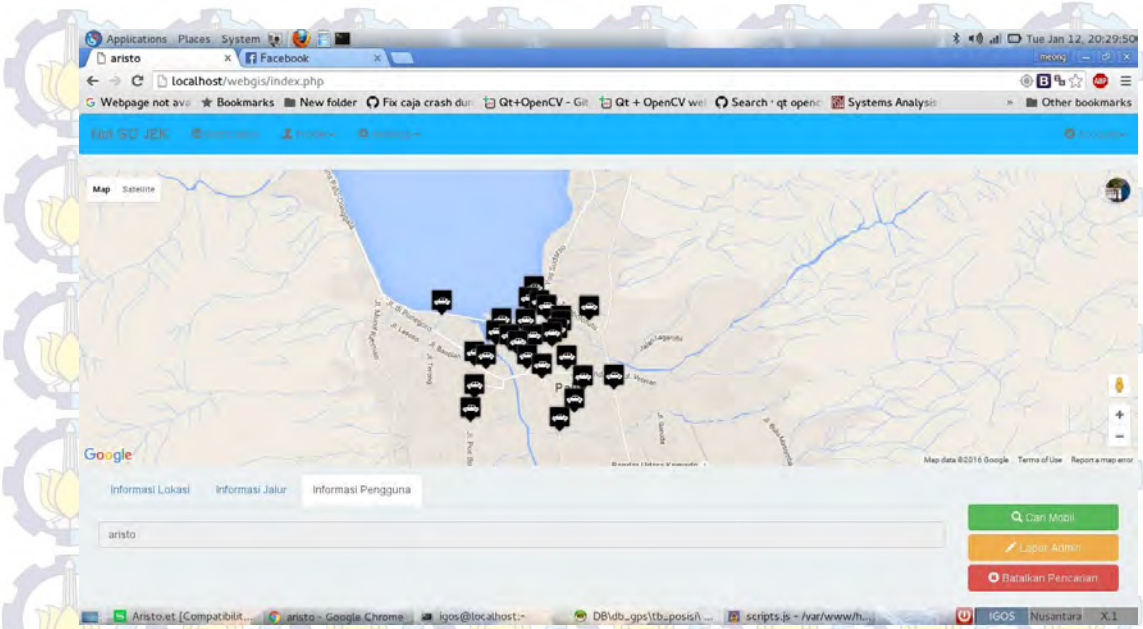
Tabel 1.3 Kalkulasi jarak antara pelanggan dengan tiap pengemudi

Pengambilan Keputusan		
Simpul Penumpang	**Simpul Mobil**	**Jarak(m)**
P1	M1	1926
	M2	442
	M3	6781
	M4	989
	M5	604
	M6	6781
Mobil Terdekat Pada Jalur Penumpang		**442**

Berdasarkan matriks ketetanggaan yang telah dibentuk, maka sistem akan mampu menentukan taksi dengan posisi yang paling dekat menuju pelanggan. Langkah selanjutnya dari sistem adalah mengirimkan pesan notifikasi yang berisi posisi dari pelanggan kepada pengemudi taksi.

1.3 Pengukuran Waktu Pencarian

Kecepatan waktu dari pencarian merupakan sebuah parameter yang menentukan kualitas dari sistem yang telah dibangun. Pengukuran dilakukan melauli proses simulasi dengan menambah jumlah node pengendara pada sistem untuk mengetahui seberapa cepat Algoritma Dijkstra yang diterapkan pada sistem yang dibangun dapat melakukan pencarian taksi terdekat bagi calon penumpang.

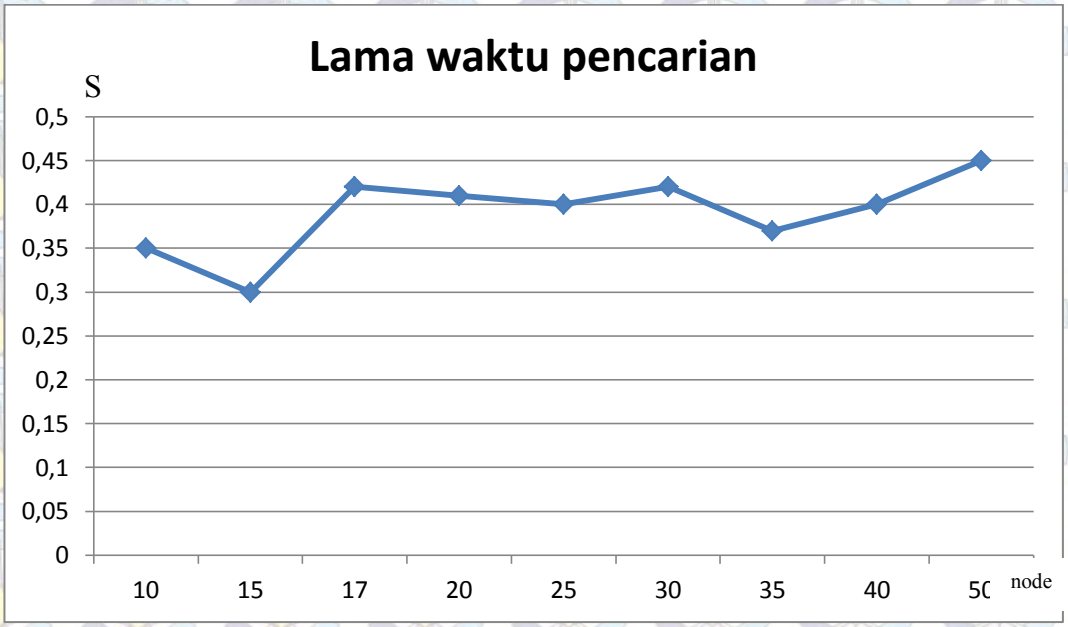


Gambar 1.7 Simulasi pengukuran waktu pelayanan

Dengan menaikkan jumlah node secara bertahap, dimulai dari 10, 20, 30, hingga berjumlah 50, dapat diketahui seberapa cepat Algoritma Dijkstra yang diterapkan pada sistem mampu bekerja mencari node terdekat bagi node pelanggan dengan asumsi bahwa sebuah perusahaan jasa angkutan umum memiliki sebuah armada yang besar dan terdiri dari banyak kendaraan. Hasil dari proses yang dilakukan dapat dilihat pada Tabel 4.4.

Tabel 1.4 Hasil pengukuran waktu pencarian

No	Jumlah node	Lama pencarian (detik)
1.	10	0,35
2.	15	0,3
3	17	0,42
4	20	0,41
5	25	0,4
6	30	0,42
7	35	0,37
8	40	0,4
9	50	0,45



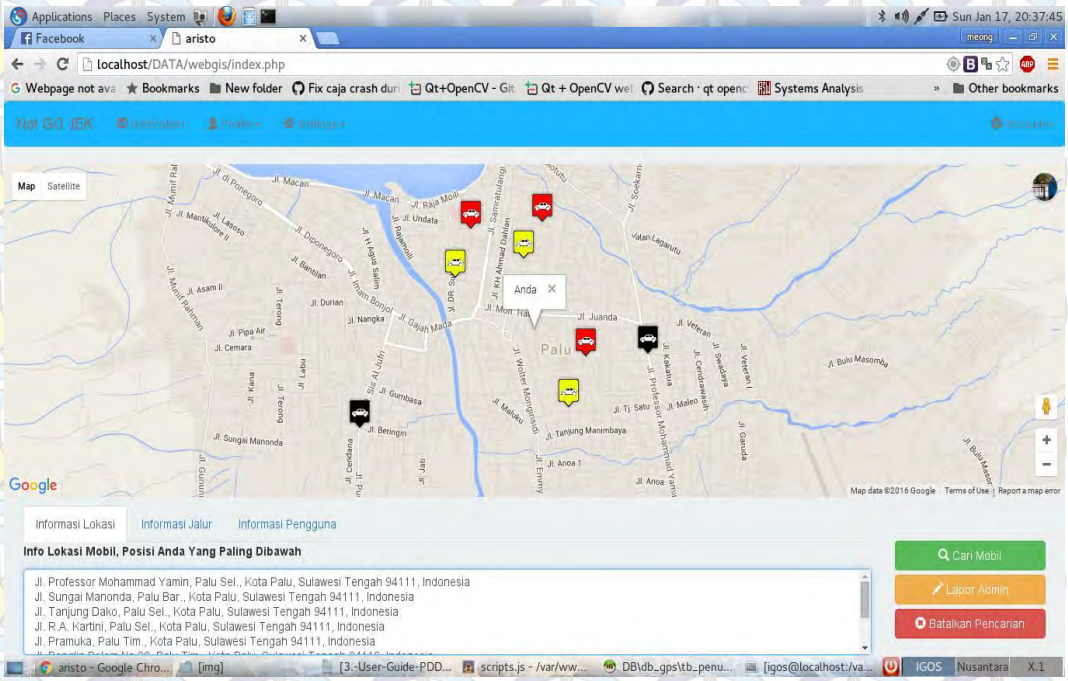
Gambar 1.8 Grafik lama waktu pencarian yang dibutuhkan oleh sistem

Melalui grafik yang ditampilkan pada Gambar 4.8, dapat dilihat bahwa meskipun waktu yang diperlukan untuk melakukan pencarian taksi terdekat cenderung fluktuatif, namun dapat dianalisa bahwa dengan semakin bertambahnya jumlah node yang ada maka waktu yang diperlukan untuk mendapatkan node yang tepat juga menjadi lebih panjang. Dalam hal ini, jika jumlah armada suatu perusahaan taksi bernilai cukup besar, maka waktu yang diperlukan oleh sistem untuk mencari taksi dengan kriteria terbaik kepada pelanggan juga akan lebih panjang. Meskipun terjadi pertambahan jumlah waktu yang diperlukan untuk mencari node yang terbaik, namun dengan melihat waktu yang diperlukan oleh sistem pada Tabel 4.4 dimulai ketika jumlah node yang masih berjumlah 10 unit lalu dinaikkan secara bertahap hingga mencapai 50 unit, maka penambahan node yang dilakukan pada proses simulasi tidak memberikan perbedaan waktu yang cukup signifikan

1.4 Simulasi kepadatan lalu-lintas

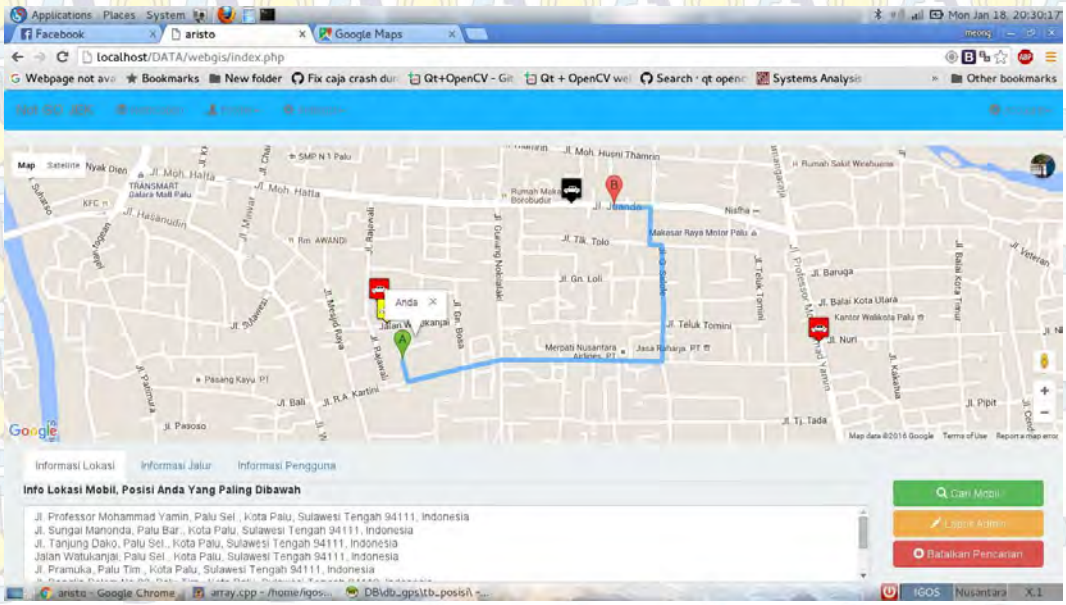
Seperti yang telah dijabarkan pada Bab 2, penggunaan hanya satu parameter yaitu jarak pada sebuah lingkungan sistem informasi geografis tidaklah memadai jika digunakan untuk pencarian sebuah rute yang valid. Hal ini disebabkan faktor selain jarak yang dapat pula digunakan sebagai acuan dalam

menentukan sebuah jalur yang valid, seperti tingkat kepadatan suatu jalan, waktu, dan berbagai hambatan lainnya. Oleh karena itu pada skenario berikut dilakukan proses simulasi yang menggambarkan pencarian node atau taksi terbaik pada pelanggan berdasarkan pada parameter jarak dan tingkat kepadatan lalu-lintas yang terjadi antara tiap-tiap node pengendara dengan node pelanggan. Pada proses simulasi ini digunakan Algoritma MDSP (*Modified Dijstra's Shortest Path*) yang merupakan pengembangan atau modifikasi dari Algoritma Dijkstra dan dapat mengakomodasi penggunaan lebih dari satu parameter dalam menentukan sebuah rute yang valid.



Gambar 1.9 Simulasi pencarian node terbaik pada kondisi kepadatan lalu-lintas yang beragam.

Skenario simulasi yang dilakukan dapat dilihat pada Gambar 4.9 dimana node yang ditandai dengan warna merah merupakan node yang terjebak dalam kondisi macet total, node kuning merupakan node yang berada dalam kondisi lalu-lintas padat namun masih dapat bergerak namun dalam kecepatan yang lambat, serta node berwarna hitam yang berada pada kondisi lalu-lintas yang lancar. Hasil dari proses simulasi yang dilakukan dapat dilihat pada Gambar 4.10 berikut.



Gambar 1.10 Hasil pencarian node terdekat dengan menggunakan Algoritma MDSP

Berdasarkan Gambar 4.10, dapat dilihat bahwa tidak jauh dari pelanggan yang memesan, sesungguhnya terdapat beberapa taksi dengan jarak yang cukup dekat. Namun dengan melakukan simulasi bahwa jalan yang mengarah menuju pelanggan tersebut tidak dapat dilewati, maka sistem memilih taksi lain yang berada pada posisi yang berada lebih jauh yang dalam simulasi ini taksi yang dipilih berada di Jalan Juanda Kota Palu yang berjarak 1,5 Km dari pelanggan yang memesan di Jalan Watukanjai Kota Palu dengan rute yang dipilih yaitu : Jl. Juanda - Jl. G. Sidole – Jl. R.A. Kartini – Jl. G. Bosa – Jl. Watukanjai.

BAB V
PENUTUP

Pada bab ini akan diuraikan beberapa kesimpulan yang dapat diambil dari pembahasan sebelumnya dan saran mengenai masalah yang bisa dibahas sebagai kelanjutan dari penelitian ini.

1.1 Kesimpulan

1. Algoritma Dijkstra yang diterapkan pada sistem yang dibangun mampu melakukan pencarian node atau taksi dengan kriteria terbaik dengan waktu rata-rata 0,3911 detik.
2. Jumlah node atau taksi yang ada akan mempengaruhi seberapa cepat sistem yang dibangun mampu melakukan pencarian node atau taksi yang paling tepat bagi pelanggan.
3. Penambahan parameter tingkat kepadatan selain parameter jarak yang telah digunakan sebelumnya serta penggunaan Algoritma MDSP yang merupakan modifikasi dari Algoritma Dijkstra pada sistem yang dibangun mampu memberikan hasil yang lebih valid dalam pencarian node taksi.

1.2 Saran

1. Perlu dilakukannya penelitian lanjutan yang menguji pemanfaatan Algoritma Dijkstra bukan hanya di sisi pengguna namun juga juga di sisi pengemudi taksi untuk meningkatkan kualitas layanan sistem terutama untuk pencarian jalur alternatif jika jalur terdekat dari taksi menuju penumpang tidak dapat dilalui.
2. Perlunya akses dengan layanan pemantau kepadatan lalu lintas agar sistem dapat melayani taksi bagi penumpang tidak hanya berdasarkan pada parameter jarak antara pengemudi taksi dengan penumpang saja naumun dapat disesuaikan dengan parameter kepadatan jalur lalu-lintas di daerah perkotaan.

3. Perlu dibangunnya aplikasi di sisi pengguna yang dapat digunakan oleh pengguna *smartphone* berbasis Android, Windows, ataupun IOS agar pengguna tidak perlu mengakses dan menerima notifikasi melalui *website* dan pesan SMS. Hal ini berguna untuk meningkatkan kecepatan layanan bagi sistem yang dibangun.

DAFTAR PUSTAKA

[1] Almomani, I.M., *et al*. (2011), "Ubiquituous GPS Vehicle Tracking and Management System", *IEEE Jordan Conferences on Applied Electrical Engineering and Computing Technologies (AEECT)*.

[2] Aloquili, O., Elbanna, A., Al-Azizi, A. (2009), "Automatic Vehicle Location Tracking System Based on GIS Environment", *The Institution of Engineering and Technology*, Vol. 3, No. 4, hal. 255-263.

[3] Cho H.J., & Cho M., (2013), "Effective Position Tracking Using B-Spline Surface Equation Based on Wireless Sensor Networks and Passive UHF-RFID", *IEEE Transactions on Instrumentation and Measurement*, Vol. 62, No. 9.

[4] Dat, H. P., Drieberg, M., Cuong, C.N. (2013), "Development of Vehicle Tracking System using GPS and GSM Modem". *IEEE Conference on Open Systems*, Sarawak.

[5] Davis, S. (2007), *GIS for Web Developers*, The Pragmatic Programmers.

[6] Gintoro, Iwan Wijaya Suharto, *et al*. (2010), "Analisis dan Perancangan Sistem Pencarian Taksi Terdekat Dengan Pelanggan Menggunakan Layanan Berbasis Lokasi". *Seminar Nasional Aplikasi Teknologi Informasi*, Yogyakarta.

[7] Jerath K., & Brennan, N. (2012), "GPS Free Train-Based Vehicle Tracking on Road Networks", *American Control Confrence*, Faimont Queen Elizabeth, Montreal.

[8] Johnson, M.C., & Thomas, L.E. (2000), *Automatic Vehicle Location Successful Transit Aplications*, U.S. Department of Transportations, Washington, DC.

[9] Joyner, D., Nguyen V.H., Cohen, N. (2011), *Algorithmic Graph Theory*.

[10] Lammle, T., (2007), *Cisco Certified Network Associate Study Guide*, Wiley Publishing, Inc., Indianapolis.

[11] Lee, S., Tewolde, G., Kwon, J., (2014), "Design and Implementation of Vehicle Tracking System Using GPS/GSM/GPRS Technology and Smartphone Application", *IEEE World Forum on Internet of Things*.

[12] McWilliam, N., *et al*. (2005), *GIS, GPS and Remote Sensing*, The Expedition Advisory Centre Royal Geographical Society 1 Kensington Gore, London.

[13] Menard, T., *et al*. (2011), "Comparing The GPS Capabilities of The Samsung Galaxy S, Motorola Droid X, and The Apple iPhone for Vehicle Tracking Using Freesim_Mobile", *International IEEE Conference on Intelligent Transportation System*, Washington, DC.

[14] Menard, T., & Miller, J., (2011), "Comparing The GPS Capabilities of The iPhone 4 and iPhone 3G for Vehicle Tracking Using FreeSim_Mobile", *IEEE Iintelligent Vehicles Symposium*, Baden-Baden.

[15] Odom, W., (2007), *CCENT/CCNA ICND1 Official Exam Certification Guide*, Cisco Press, Indianapolis.

[16] Sathiamoorthy, M. (2009), "On GPS Tracking of Mobile Devices", *International Conference on Networking and Services*".

[17] Sivakumar, S., & C. Chandrasekar. (2014), Modified Dijkstra's Shortest Path Algorithm", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 2.

[18] Yi Z., Ding W., Yanfei Y. (2009), "On Dynamic Scheduling of Vehicles Based on GPS / GIS / RFID", *Department of Computer Science & Technology*, Harbin.

[19] Yuan G., *et al*. (2008), "Research and Design of GIS in Vehicle Monitoring System", *Internation Conference of Internet Computing in Science and Engineering*.

LAMPIRAN

7.1 Kode Program Server

```
$(document).ready(function(){
	var appName =navigator.appVersion;
	var cekAppName = appName.split(" ");

	setInterval(function(){
		window.location.reload();
	}, 500000);
	//ambil nama pengguna
	$.ajax(
	{
	 url : "login.php?action=getUser",
	 type: "POST",
	 success:function(data, textStatus, jqXHR)
		{
			document.getElementById("userPengguna").value = data;
		},
	 error: function(jqXHR, textStatus, errorThrown)
		{
		 alert(errorThrown);
		}
	});
	if (cekAppName[2] == 'Android' || cekAppName[2] == 'android' ||
cekAppName[2] == 'Iphone' || cekAppName[2] == 'iphone'){
		$("#panelJalan").hide();
		$("#tabs-662392").hide();
		$("#lprAdmin").hide();
	}
```

```
        else{
                $("#panelJalan").show();
        }

        $("#suNav").click(function(){
                $("#settingMaps").modal('show');
        });

        $("#smNav").click(function(){
                $("#settingUmum").modal('show');
        });

        $("#").click(function(){
        });

        $("#").click(function(){
        });
});

function tampilSettingsUmum(){

}
var  data = ambilAjax("latlong.php?action=latlong",'get',false);
google.maps.event.addDomListener(window, 'load', initMap);

var mobilDB = JSON.parse(data);

function cariJalur(){  \\Graf
        var dataJalan =
ambilAjax("getJarak.php?action=dataDijkstra&idAmbil="+document.getElement
ById("idPengguna").value,'get',false);
```

```
        var dataPosisiPengguna =
ambilAjax("getJarak.php?action=posisiSaya&idTerakhir="+document.getElement
ById("idPengguna").value,'get',false);
        var obj = jQuery.parseJSON(dataJalan);
        var objPengguna = jQuery.parseJSON(dataPosisiPengguna);
        for (var q=0; q<=obj.length;q++){
            if (q<6){

        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
            }
            else if(q<12){

        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
            }
            else if(q<19){

        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
            }
            else if(q<25){

        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
            }
            else if(q<31){

        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
            }
            else if(q<37){

        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
            }
            else if(q<39){
```

```
        makeRoads(obj[q].start,obj[q].finish,parseInt(obj[q].jarak));
                }

        }
        var jarMin=[];
        var hitung=[];
        var jarMin1;
        for (var aw=0; aw<obj.length;aw++){
                if (aw>=0 && aw<6){
                        jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                        if(jarMin1[0] != 0){
                                if(jarMin[0] == null){
                                        jarMin[0] = jarMin1;
                                }
                                else{
                                        if(jarMin[0][0] > jarMin1[0] && jarMin[0]
!= 0){
                                                jarMin[0] = jarMin1;
                                        }
                                }
                        }

                }
                else if(aw>=6 && aw<12){
                        jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                        if(jarMin1[0] != 0){
                                if(jarMin[1] == null){
                                        jarMin[1] = jarMin1;
                                }
```

```
                    else{
                        if(jarMin[1][0] > jarMin1[0] && jarMin[1]
!= 0){
                            jarMin[1] = jarMin1;
                        }
                    }
                }
            }
            else if(aw>=12 && aw<19){
                jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                if(jarMin1[0] != 0){
                    if(jarMin[2] == null){
                        jarMin[2] = jarMin1;
                    }
                    else{
                        if(jarMin[2][0] > jarMin1[0] && jarMin[2]
!= 0){
                            jarMin[2] = jarMin1;
                        }
                    }
                }
            }
            else if(aw>=19 && aw<25){
                jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                if(jarMin1[0] != 0){
                    if(jarMin[3] == null){
                        jarMin[3] = jarMin1;
                    }
                    else{
```

```
                        if(jarMin[3][0] > jarMin1[0] && jarMin[3]
!= 0){
                            jarMin[3] = jarMin1;
                        }
                    }
                }
            else if(aw>=25 && aw<31){
                jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                if(jarMin1[0] != 0){
                    if(jarMin[4] == null){
                        jarMin[4] = jarMin1;
                    }
                    else{
                        if(jarMin[4][0] > jarMin1[0] && jarMin[4]
!= 0){
                            jarMin[4] = jarMin1;
                        }
                    }
                }
            }
            else if(aw>=31 && aw<37){
                jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                if(jarMin1[0] != 0){
                    if(jarMin[5] == null){
                        jarMin[5] = jarMin1;
                    }
                    else{
                        if(jarMin[5][0] > jarMin1[0] && jarMin[5]
!= 0){
```

```
                                    jarMin[5] = jarMin1;
                                }
                            }
                        }
                    }
                    else if(aw>=37 && aw<42){
                        jarMin1=Dijkstra(roads, objPengguna[0].posisi,
obj[aw].finish);
                        if(jarMin1[0] != 0){
                            if(jarMin[6] == null){
                                jarMin[6] = jarMin1;
                            }
                            else{
                                if(jarMin[6][0] > jarMin1[0] && jarMin[6]
!= 0){
                                    jarMin[6] = jarMin1;
                                }
                            }
                        }
                    }
                }
                //console.log(hitung);
                //console.log(jarMin)
                //var sa = Dijkstra(roads, objPengguna[0].posisi, obj[10].finish);
                var palingKecil;
                for (var ab =0; ab<jarMin.length;ab++){
                    if(ab == 0){
                        palingKecil = jarMin[ab];
                    }
                    else{
                        if(palingKecil[0] > jarMin[ab][0]){
                            palingKecil = jarMin[ab];
```

```
                }
            }
        }
        console.log(palingKecil);
        return palingKecil;
}

function ambilAjax(url,type,async)
{
    var result = null;
    $.ajax({
      url: url,
      type: type,
      async: async,
      success: function(data) {
        result = data;
      }
    });
    return result;
}

function kirimData(url,d){
        $.ajax({
            url: url,
            type: 'POST',
            data:d,
            success: function(status) {
                //console.log(status);
            }
        });
}
```

```
function initMap() {
  var directionsService = new google.maps.DirectionsService;
  var directionsDisplay = new google.maps.DirectionsRenderer;
  var map = new google.maps.Map(document.getElementById('map'), {
   center: {lat: -0.8193728, lng: 119.8722506},
   zoom: 10
  });
  var infoWindow = new google.maps.InfoWindow({map: map});
  var icon='img/car.png';
  if (navigator.geolocation) {
   navigator.geolocation.getCurrentPosition(function(position) {
    var pos = {
     lat: position.coords.latitude,
     lng: position.coords.longitude
    };

    infoWindow.setPosition(pos);
    infoWindow.setContent('Anda');
    map.setCenter(pos);
    map.setZoom(14);
    directionsDisplay.setMap(map);
    directionsDisplay.setPanel(document.getElementById('panelJalan'));

    var objTempat=titikJalan(mobilDB);
    objTempat.push(pos);
    var objTujuan=titikJalan(mobilDB);
    objTujuan.push(pos);
    var geocoderSaya = new google.maps.Geocoder;
    var geocoderMobil = new google.maps.Geocoder;
    tambahMarker(mobilDB,map);
        buatJalan(objTempat,objTujuan);
        posisiSaya(pos);
```

```
            tampilPosisiMobil(geocoderMobil,mobilDB);

            var onClick = function() {
                var a = cariJalur();
                cariIdOto(a);
                console.log(a);
                if (a==false){
                        alert("Tidak Ada Jalan");
                }
                else{
                        tampilJalan(directionsService,directionsDisplay,a[1],a[2]);
                }
            };
            document.getElementById('cariMobil').addEventListener('click',
    onClick);

      },function() {
        handleLocationError(true, infoWindow, map.getCenter());
      });
    } else {
      handleLocationError(false, infoWindow, map.getCenter());
    }
  }

  function handleLocationError(browserHasGeolocation, infoWindow, pos) {
    infoWindow.setPosition(pos);
    infoWindow.setContent(browserHasGeolocation ?
                          'Error: Geo Lokasi Tidak Ditemukan.' :
                          'Error: Browser Anda Tidak Mendukung.');
  }

  function tambahMarker(location, map) {
```

```
  var marker;
  var icon='img/car.png';
  var markers = new Array();
  for (var i = 0; i < location.length; i++) {
                marker = new google.maps.Marker({
                        position: new google.maps.LatLng(mobilDB[i].lat,
mobilDB[i].long),
                        map: map,
                        icon: icon
                });
                markers.push(marker);
                google.maps.event.addListener(marker, 'click', (function(marker,
i) {
                        return function() {
                         infowindow.setContent(locations[i][0]);
                         infowindow.open(map, marker);
                        }
                })(marker, i));
        }
}
var titikJalan=function(location){
        var xy=[];
        for (var i = 0; i < location.length; i++) {

        xy.push({lat:parseFloat(location[i].lat),lng:parseFloat(location[i].long)});
        }
        return xy;
};

function buatJalan(tempat,mobil){
        var service = new google.maps.DistanceMatrixService;
```

```
service.getDistanceMatrix({
        origins: tempat,
        destinations: mobil,
        travelMode: google.maps.TravelMode.DRIVING,
        unitSystem: google.maps.UnitSystem.METRIC,
        avoidHighways: false,
        avoidTolls: false
},
function(response, status){
        var originList = response.originAddresses;
        var destinationList = response.destinationAddresses;
        var jarak = [];
        var user = document.getElementById("userPengguna").value;
        //alert(user);
        if (status !== google.maps.DistanceMatrixStatus.OK) {
                alert('Error was: ' + status);
        }
        else {
                for (var i = 0; i < originList.length; i++) {
                        var results = response.rows[i].elements;
                        for (var j = 0; j < results.length; j++) {
                                jarak.push({

id:document.getElementById("idPengguna").value+i+j,

pengguna:document.getElementById("idPengguna").value,
                                        asal:originList[i],
                                        tujuan: destinationList[j],
                                        jarak:results[j].distance.value
                                });
                        }
                }
```

```
                    for (var d = 0; d < jarak.length; d++) {
                        if (jarak[d].tujuan == jarak[d].asal){
                            jarak.splice(d,1);
                        }

        kirimData("getJarak.php?action=getJarak",jarak[d]);
                    }
                    $.each(jarak,function(data){
                        var x = document.getElementById("jalurLokasi");
                        var option = document.createElement("option");
                        option.text = "Asal :"+this.asal+',Tujuan
:'+this.tujuan+",jarak :"+this.jarak;
                        x.add(option);
                    });

                    $.each(destinationList,function(data){
                        var x = document.getElementById("infoLokasi");
                        var option = document.createElement("option");
                        option.text = this;
                        x.add(option);
                    });
                }
        });
}

function posisiSaya(pos) {
  var service = new google.maps.DistanceMatrixService;
        var latlng=[pos];
        service.getDistanceMatrix({
                origins: latlng,
                destinations: latlng,
                travelMode: google.maps.TravelMode.DRIVING,
```

```
            unitSystem: google.maps.UnitSystem.METRIC,
            avoidHighways: false,
            avoidTolls: false
        },

        function(response, status){
            var originList = response.originAddresses;
            var alamat =[];
            var alamatPenumpang=[];
            if (status !== google.maps.DistanceMatrixStatus.OK) {
                alert('Error was: ' + status);
            }
            else {
                for (var i = 0; i < originList.length; i++) {
                    var results = response.rows[i].elements;
                    alamat.push({
                        penumpang:
document.getElementById("idPengguna").value,
                        latitude:latlng[i].lat,
                        longitude:latlng[i].lng,
                        posisiTerakhir:originList[i]
                    });

        kirimData("getJarak.php?action=getPosisiPenumpang",alamat[i]);
                }
            }
        });
}

function tampilPosisiMobil(geocoder,pos)
{
```

```
        var service = new google.maps.DistanceMatrixService;
        var latlng=[];
        for (var a=0; a<pos.length;a++){
                latlng.push({lat: parseFloat(pos[a].lat), lng:
parseFloat(pos[a].long)});
        }
        service.getDistanceMatrix({
                origins: latlng,
                destinations: latlng,
                travelMode: google.maps.TravelMode.DRIVING,
                unitSystem: google.maps.UnitSystem.METRIC,
                avoidHighways: false,
                avoidTolls: false
        },

        function(response, status){
                var originList = response.originAddresses;
                var alamat =[];
                var kirimAlamatTerakhir=[];
                if (status !== google.maps.DistanceMatrixStatus.OK) {
                        alert('Error was: ' + status);
                }
                else {
                        for (var i = 0; i < originList.length; i++) {
                                var results = response.rows[i].elements;
                                alamat.push({
                                        posisiTerakhir:originList[i]
                                });
                        }
                }
                for (var a=0; a<pos.length;a++){
                        kirimAlamatTerakhir.push({
```

```
                                    mobil:pos[a].id,
                                    jalan:alamat[a].posisiTerakhir
                                });

        kirimData("getJarak.php?action=getPosisiTerakhir",kirimAlamatTerakhir[
a]);
                        }

                    });
}

function tampilJalan(directionsService,directionsDisplay,posisi,tujuan) {
  directionsService.route({
   origin: posisi,
   destination: tujuan,
   travelMode: google.maps.TravelMode.DRIVING
  }, function(response, status) {
          var legs = directionsService.durationInTraffic;
   if (status === google.maps.DirectionsStatus.OK) {
                directionsDisplay.setDirections(response);
                 $("#modalPenunjuk").modal('show');
   } else {
     window.alert('Directions request failed due to ' + status);
   }
  });
}
//algoritma dijkstra
function Dijkstra(roads, source, dest) {
  var inf = Number.POSITIVE_INFINITY;
  var distance = {};
  var done = {};
  var pred = {};
```

```
for (var i in roads) {
 // jalan tidak diketahui
 distance[i] = inf;
 pred[i] = 0;
 done[i] = false;
}

// jalan dari tempat to ketujuan = 0
distance[source] = 0;

for (i in roads) {
 var minDist = inf, closest;
 for (var j in roads) {
  if (!done[j]) {
   if (distance[j] <= minDist) {
    minDist = distance[j];
    closest = j;
   }
  }
 }
 done[closest] = true;
 if (closest === dest) {
  break;
 }

 var neighbors = roadsFrom(closest);
 for (var nb in neighbors) {
  var w = neighbors[nb];
  if (!done[nb]) {
   if (distance[closest] + w < distance[nb]) {
    distance[nb] = distance[closest] + w;
    pred[nb] = closest;
```

```
      }
     }
    }
   }

   // Selesai, cetak jalur.
   i = dest;
   if (distance[i] < inf) {
    var thePath = i;
    var place = i;
    while (place !== source) {
     place = pred[place];
     if (place !== source) {
      thePath = place + '->' + thePath;
     }
    }
    thePath = place + '->' + thePath;
    return [distance[i],source,dest];
   } else {
    return false;
   }
  }

  var roads = {};
  function makeRoad(from, to, length) {
   function addRoad(from, to) {
    if (!(from in roads)) {
     roads[from] = {};
    }
    roads[from][to] = length;
   }
   addRoad(from, to);
```

```
  addRoad(to, from);
 }

 function makeRoads(start) {
  for (var i = 1; i < arguments.length; i += 2) {
   makeRoad(start, arguments[i], arguments[i + 1]);
  }
 }

 function roadsFrom(place) {
  var found = roads[place];
  if (found === undefined) {
   console.log("No place named '" + place + "' found.");
  } else {
   return found;
  }
 }

 function cariIdOto(dataJalan){
        $.get("isiOrder.php?action=cariIdMobil&alamat="+dataJalan[2],
 function(data){
     var obj = JSON.parse(data);
     for (var a=0; a<obj.length;a++){
                    kirimPesan(obj[a].id,"1 Order Diterima");
                }
  });
 }

 function kirimPesan(idMobil,isiPesan){
        $("#btnPesanMobil").click(function(){
                simpanOrder(idPengguna,idMobil);
```

```
        $.get("isiOrder.php?action=kirimPesan&pesan="+isiPesan+"&idMobil="+
idMobil, function(data){
                        console.log("Data: " + data + "\nStatus: " + status);
                        setInterval(function(){
                                window.location.reload();
                        },80000);
                });
        });
}

function simpanOrder(idPengguna,idMobil){
        $.get("isiOrder.php?action=simpanOrder&pengguna="+idPengguna+"&m
obil="+idMobil, function(data){
                $("#pesananDialog").modal('show');
        });
}
```

7.2 Kode Program GPS Tracker

```
package id.flwi.example.gcmappdemo;
import java.io.IOException;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.BasicResponseHandler;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class GpsActivity extends Activity {

    GpsService      gps;
    Button btnUpdate;
    Button btnCariLokasi;
    EditText noPol;
    EditText namaSupir;
    EditText latid;
    EditText longit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gps);

        btnUpdate = (Button) findViewById(R.id.btnUpdate);
        btnCariLokasi = (Button) findViewById(R.id.btnCariLokasi);

        noPol = (EditText) findViewById(R.id.editNopol);
        namaSupir = (EditText) findViewById(R.id.editSopir);
        latid = (EditText) findViewById(R.id.editLatitude);
        longit = (EditText) findViewById(R.id.editLongitude);

        noPol.setKeyListener(null);
        //namaSupir.setKeyListener(null);
        latid.setKeyListener(null);
```

```
longit.setKeyListener(null);

btnCariLokasi.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View arg0) {
                gpsAct();
        }
});

btnUpdate.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View arg0) {
                String plat = noPol.getText().toString();
                String lat=latid.getText().toString();
                String longits =longit.getText().toString();

                String
url="http://103.245.72.45/webgis/uploadLokasi.php?action=updateLokasi&id
mobil="+plat+"&latitude="+lat+"&longitude="+longits+"&speed=0&status=
AUTOLOW";
                new actionUpload().execute(url);
        }
});

// buat class object dari GpsService
gps = new GpsService(GpsActivity.this);

// dicek dulu apakah GPSnya idup
if (gps.canGetLocation())
{
```

```
                // ambil latitude dan longitude
                double latitude = gps.getLatitude();
                double longitude = gps.getLongitude();

                String isiLat = Double.toString(latitude);
                String isiLong = Double.toString(longitude);

                latid.setText(isiLat);
                longit.setText(isiLong);
                noPol.setText("357671031485634");
                namaSupir.setText("Aristo Tab");

        } else
        {
                gps.showSettingAlert();
        }

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.gps, menu);
        return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
```

```
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    public void gpsAct(){
        gps = new GpsService(GpsActivity.this);

        if (gps.canGetLocation())
        {
            // ambil latitude dan longitude
            double latitude = gps.getLatitude();
            double longitude = gps.getLongitude();
            String isiLat = Double.toString(latitude);
            String isiLong = Double.toString(longitude);
            AlertDialog.Builder viewLoc = new
AlertDialog.Builder(GpsActivity.this);
            viewLoc.setTitle("Lokasi Anda Saat Ini");
            viewLoc.setMessage("Latitud :"+isiLat+"\n Longitude
:"+isiLong);
            viewLoc.setNeutralButton("OK", null);
            viewLoc.show();
        } else
        {
            gps.showSettingAlert();
        }
    }

    private class actionUpload  extends AsyncTask<String, Void, Void> {

      private final HttpClient Client = new DefaultHttpClient();
```

```
        private String Content;
        private String Error = null;
        protected void onPreExecute() {

        }

        // Call after onPreExecute method
        protected Void doInBackground(String... urls) {
          try {
            HttpGet httpget = new HttpGet(urls[0]);
            ResponseHandler<String> responseHandler = new
BasicResponseHandler();
            Content = Client.execute(httpget, responseHandler);

          } catch (ClientProtocolException e) {
            Error = e.getMessage();
            cancel(true);
          } catch (IOException e) {
            Error = e.getMessage();
            cancel(true);
          }

          return null;
        }

        protected void onPostExecute(Void unused) {
          if (Error != null) {
             Toast.makeText(GpsActivity.this, "Lokasi Berhasil Diupdate",
Toast.LENGTH_LONG).show();
          }
          else{
```

```
                Toast.makeText(GpsActivity.this, "Lokasi Tidak Berhasil
Diupdate", Toast.LENGTH_LONG).show();
            }
        }

    }
}
```

BIOGRAFI PENELITI

Muh. Aristo Indrajaya, lahir di Kota Palu, Provinsi Sulawesi Tengah pada tanggal 10 Agustus 1990. Putra pertama dari dua bersaudara dari pasangan Hasan Basri dan Syamsiar menyelesaikan pendidikan Strata 1 di Universitas Tadulako, Palu, Sulawesi Tengah pada tahun 2013.

***Peneliti dapat dihubungi pada email* :** aristocool@yahoo.com