BAB II
TEORI PENUNJANG

Pada bab ini akan diuraikan mengenai teori dan perangkat penunjang yang digunakan pada perancangan dan implementasi dari sistem perancangan *middleware*. Pembahasan berisi tentang *Smart Grid, middleware,* komunikasi data, *web services*, *HyperText Transfer Protocol, Uniform Resource Identifier*, *Dynamic Domain Name System, JavaScript Object Notation* dan *Wireshark*.

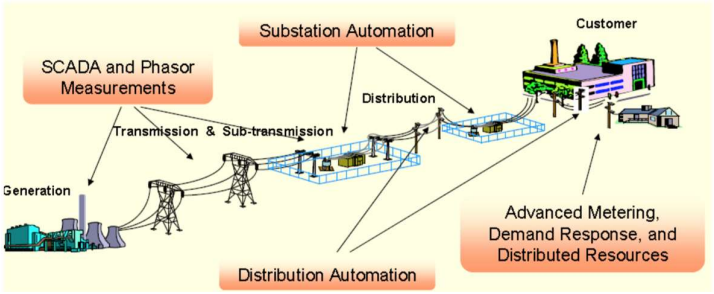
2.1 *Smart Grid*

Smart Grid adalah suatu jaringan listrik yang menggunakan teknologi digital dan teknologi maju lainnya untuk memantau dan mengelola transportasi listrik dari sumber pembangkitan listrik untuk memenuhi perubahan kebutuhan listrik dari pelanggan[2]. Suatu *Smart Grid* secara cerdas mengintegrasikan kegiatan semua *user* dalam rangka memberikan atau menghasilkan suplai listrik secara efisien, berkesinambungan, ekonomis dan aman.

Smart Grid memakai produk inovatif dan bersama-sama melayani dengan pemantauan, pengendalian, komunikasi dan *self-healing technologies* untuk:

a. Memfasilitasi lebih baik hubungan dan operasi dari semua generator dan teknologi.
b. Memberikan keleluasaan kepada pelanggan untuk menentukan bagian dari optimasi operasi dari sistem.
c. Menyediakan pelanggan dengan informasi dan pilihan untuk suplai.
d. Secara signifikan mengurangi dampak lingkungan dari sistem suplai listrik seluruhnya.
e. Meningkatkan kehandalan dan keamanan suplai.

Teknologi *Smart Grid* merupakan teknologi yang memanfaatkan kemajuan teknologi komunikasi, komputer dan *cyber* untuk dapat melakukan pengendalian dan pegoperasian sistem tenaga listrik dalam menyalurkan energi listrik. Oleh karena itu, *Smart Grid* merupakan integrasi teknologi cerdas dalam jaringan listrik yang dicita-citakan untuk memenuhi kebutuhan energi listrik yang di masa sekarang maupun masa mendatang menjadi kebutuhan primer.



Gambar 2.1 Desain Konseptual *Smart Grid.*

Ada pula keuntungan dari *Smart Grid*, yaitu sebagai berikut:

1. Mengurangi jumlah daya gardu yang diperlukan, karena utilitas listrik tahu persis berapa banyak jaringan listrik yang dibutuhkan pada waktu tertentu.
2. Mengintegrasikan sumber energi terbarukan ke dalam jaringan komunikasi.
3. Mengurangi beban puncak selama ada konsumsi energi.

Meskipun *Smart Grid* merupakan teknologi terbaru, tetapi di setiap sistem tentu memiliki kelemahan, diantaranya adalah kelemahan *Smart Grid*:

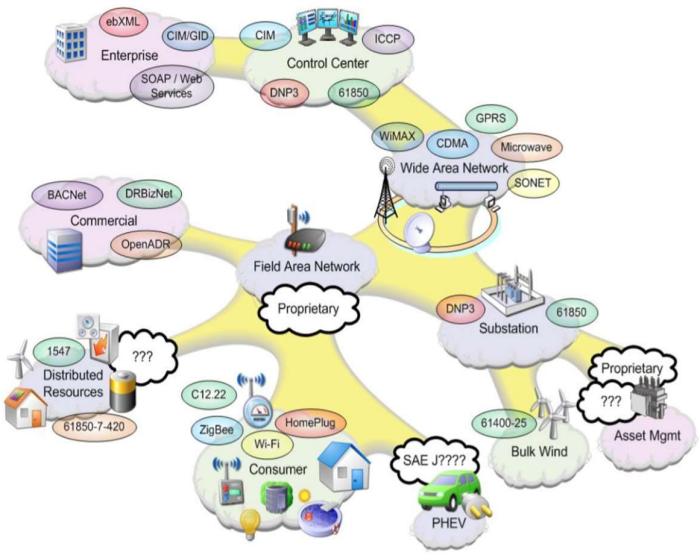
1. Ada risiko keuangan yang cukup besar.
2. Banyak perusahaan belum mau berinvestasi sampai teknologi *Smart Grid* diuji secara ekstensif.

2.1.2 Standardisasi *Smart Grid*

Pemerintah dan industry harus mengevaluasi prioritas dan menetapkan protokol, definisi, dan standar untuk peralatan, data transportasi, interoperabilitas dan keamanan dunia maya, membuat rencana untuk pengembangan standar.

Dalam membuat standar, perlu dilakukan perluasan kerja sama dalam pengembangan standar internasional untuk mengurangi biaya dan mempercepat inovasi sambil mengembangkan standar yang diterima secara global.

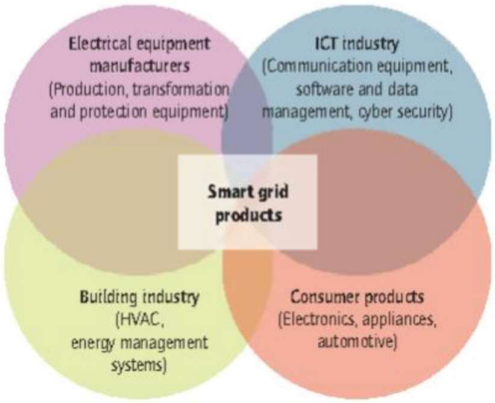
Peralatan *Smart Grid* dan sistem yang disediakan oleh banyak sektor industry yang secara historis tidak bekerja bersama, seperti produsen peralatan, penyedia ICT, industri bangunan, produk konsumen dan pemasok jasa.



Gambar 2.2 Beberapa Standar Dalam Lingkup Komunikasi Data *Smart Grid*.

Sistem kontrol yang dioperasikan oleh utilitas jaringan interkoneksi harus mampu bertukar informasi. Peralatan-peralatan cerdas milik pelanggan, sistem manajemen energi dan kendaraan listrik perlu berkomunikasi dengan *Smart Grid*.

Standar, definisi dan protokol untuk transportasi data sangat penting untuk "sistem dari sistem" yang kompleks untuk beroperasi secara mulus dan aman.



Gambar 2.3 Penyedia Produk *Smart Grid*.

Dalam penerapan sistem *Smart Grid*, diperlukan suatu standar, karena standar dapat menghindari terjadinya "*re-inventing the wheel"*, dapat belajar dari praktek-praktek terbaik industri, menetapkan persyaratan yang lebih mudah, mengurangi biaya integrasi, mencegah vendor tunggal "*lock-in"* dan vendor

berbagi pasar yang jauh lebih besar. Standar *Smart Grid* membutuhkan pengintegrasian bermacam-macam keahlian, seperti:

a. Penelitian pengukuran industri utilitas listrik.
b. Teknologi jaringan maju.
c. Bangunan dan infrastruktur kelistrikan.
d. Komputer dan keamanan jaringan.
e. Keahlian standar dokumentasi.
f. Pengalaman dalam pengujuan dan sertifikasi.

2.1.3 Pemanfaatan *Smart Grid* Untuk Perkotaan

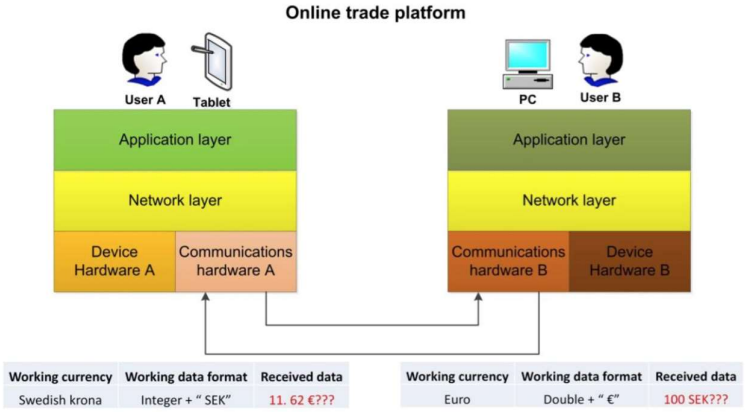
Teknologi pengendalian *Smart Micro Grid* yang telah diterapkan di Sumba untuk mengintegrasikan energi terbarukan dengan sistem tenaga listrik yang sudah ada akan diadopsi untuk diterapkan pada pengontrolan operasi sistem tenaga listrik perkotaan, terutama pada blok-blok bangunan tingga yang memerlukan keandalan dan efesiensi tinggi[3]. *Demo Plant Smart Micro Grid* yang pertama di Indonesia saat ini akan menjadi rujukan baik dari segi teknologi maupun segi keekonomiannya terhadap perkembangan *Smart Grid* yang mengintegrasikan energi terbarukan pada sistem kelistrikan di Indonesia.

Walaupun penerapan *Demo Plant Micro Smart Grid* dilakukan di sistem kelistrikan daerah, penerapan teknologi ini juga dapat dilakukan pada sistem kelistrikan kota-kota besar di Indonesia, seperti sistem Jakarta yang memiliki beban yang paling bervariasi, mulai dari rumah tangga, sektor komersial dan industri, serta kebutuhan keandalan yang tinggi dari sistem tenaga listrik dan kebebasan memilih jenis layanan listrik juga meningkat pada kota-kota besar, memperlihatkan secara teoritis bahwa aplikasi teknologi *Smart Grid* layak ditimbang untuk diterapkan.

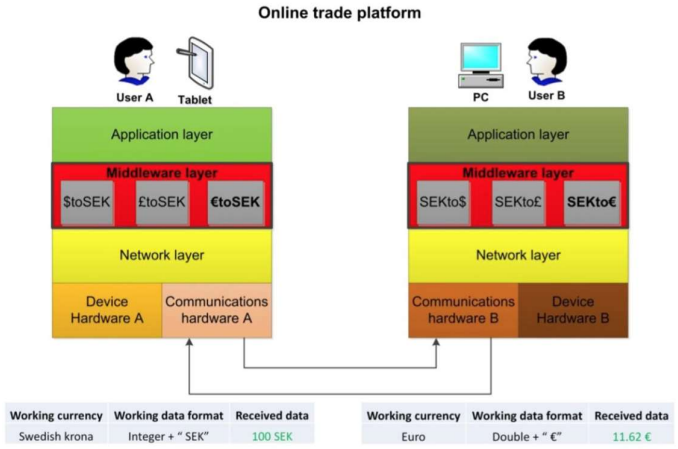
2.2 *Middleware*

Guna mengintegrasikan semua perangkat yang berbeda dengan menyediakan layanan komunikasi data, oleh karena itu, diperlukan lapisan terpadu. Lapisan ini tidak hanya memadukan semua kompleksitas dari komponen-komponen *Smart Grid* yang berfungsi untuk pengukuran cerdas, tetapi juga struktur jaringan listrik, lalu kemudian menjaga *user* dari struktur jaringan tersebut. Masalah pada lapisan harus dikembangkan, karena *middleware* dapat mengintegrasikan komponen dan

perangkat yang berbeda pada *Smart Grid* sehingga bisa menjadi satu lapisan heterogen. Seperti kasus pada Gambar 2.5, jika ada transaksi pada pasar uang *online*, kemungkinan akan terjadi kesalahan konversi bila tanpa adanya lapisan *middleware*. Pada awalnya, tidak ada masalah di lapisan bawah karena hanya menangani *Protocol Data Units* (PDU). Pengiriman informasi hanya akan terus dilanjutkan bila format PDU sudah benar. Akan tetapi, ketika konten diterima oleh lapisan aplikasi, biasanya timbul masalah interpretasi (sebagai contoh, bila informasi diterima oleh perangkat atau peralatan dengan prosesor yang berbeda atau perbedaan metode penyimpanan *byte*). Bila data ingin dikonversi dengan benar pada lapisan aplikasi, maka solusinya adalah dengan lapisan *middleware*. Bukan hanya itu saja, permasalahan mengenai format yang berbeda pun dapat teratasi seperti pada Gambar 2.5.



Gambar 2.4 Ketidakcocokan Format Data Pengirim dan Penerima.



Gambar 2.5 Meningkatnya Sistem Komunikasi Dengan Lapisan *Middleware*.

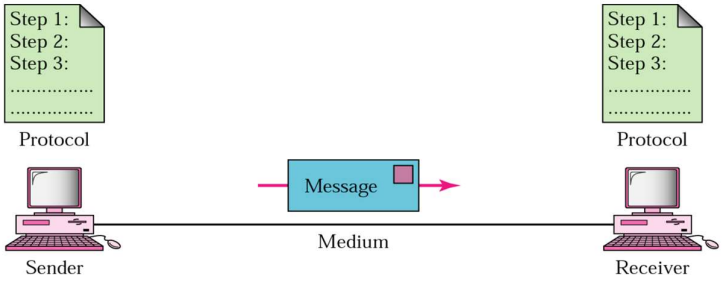
Oleh karena itu, selain *middleware* bekerja sebagai jembatan antara dua perangkat atau lebih, *middleware* juga dibutuhkan agar tidak terjadi kesalahan informasi yang diterima pada pusat kendali.

2.3 Komunikasi Data

Komunikasi data merupakan pertukaran data antara dua perangkat melalui beberapa bentuk media transmisi seperti kabel kawat. Guna melangsungkan komunikasi data, perangkat yang berkomunikasi haruslah menjadi bagian dari sistem komunikasi yang terdiri dari kombinasi perangkat keras (*hardware*) dan perangkat lunak (*software*)[3]. Keefektivan suatu sistem komunikasi data adalah tergantung dari empat sifat mendasar, yaitu sebagai berikut:

1. Pengiriman. Sistem harus mengirimkan data ke tujuan yang diinginkan. Data yang dikirimkan tersebut wajib diterima oleh perangkat yang dituju atau *user* yang dituju.
2. Akurasi. Sistem harus mengirimkan data yang akurat. Perubahan data yang terjadi dalam transmisi akan menyebabkan data menjadi tidak dapat digunakan.
3. Ketepatan waktu. Sistem harus mengirimkan data pada waktu yang tepat. Data yang terlambat dikirimkan tidak dapat digunakan.
4. *Jitter*. *Jitter* mengacu pada variasi waktu kedatangan paket. Biasanya terjadi d*elay* acak pada pengiriman paket suara atau video.

Sistem komunikasi data memiliki 5 komponen yang ditunjukkan pada Gambar 2.6.



Gambar 2.6 Lima Komponen Komunikasi Data.

1. Pesan (*Message*). Pesan adalah informasi (data) yang akan dikomunikasikan.
2. Pengirim (*Sender*). Pengirim adalah perangkat yang mengirimkan data pesan.
3. Penerima (*Receiver*). Penerima adalah perangkat yang menerima data pesan.

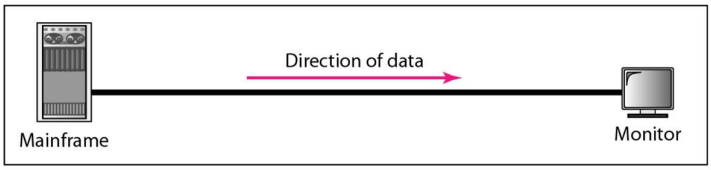
4. Media transmisi (*Medium*). Media transmisi adalah media yang digunakan untuk melakukan pengiriman data.
5. Protokol (*Protocol*). Protokol adalah aturan-aturan yang berfungsi untuk mengatur komunikasi data.

2.3.1 Alur Pertukaran Data

Alur pertukaran data pada komunikasi antara dua perangkat terbagi menjadi *simplex*, *half-duplex,* dan *full-duplex*.

1. *Simplex*

   Simplex merupakan komunikasi searah, layaknya seperti jalan raya yang memiliki jalur searah. Hanya satu yang bisa mentransmisikan data dari dua perangkat yang terhubung, sedangkan yang lainnya hanya dapat menerima data tersebut. Contohnya adalah seperti *Keyboard* bekerja melakukan *input*, sedangkan *monitor* bekerja menerima *output*. Mode *simplex* dapat memanfaatkan semua kapasistas kanal untuk mengirimkan data, tetapi hanya pengiriman searah saja.

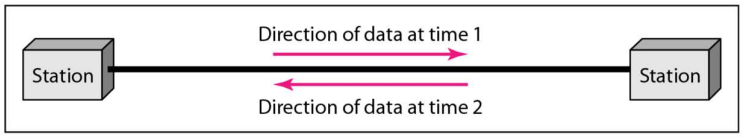


Gambar 2.7 *Simplex,* Komunikasi Searah.

2. *Half-Duplex*

   Pada mode *half-duplex*, setiap stasiun dapat melakukan transmisi dan menerima transmisi, tetapi tidak pada saat yang bersamaan. Ketika salah satu perangkat melakukan transmisi, yang lain hanya dapat menerima transmisi, dan begitu pun sebaliknya. Mode *half-duplex* seperti jalan raya yang memiliki satu jalur akan tetapi dapat direkayasa untuk jalur dua arah. Ketika kendaraan melalui jalur satu arah, kendaraan yang datang dari jalur lain harus menunggu. Dalam transmisi *half-duplex*, seluruh kapasitas saluran diambil alih oleh perangkat melakukan transmisi pada saat itu juga. *Walkie-talkie* dan CB *(citizen band),* keduanya merupakan radio yang menggunakan sistem *half-duplex*. Mode *half-duplex* biasanya digunakan dalam kasus-kasus yang mana tidak diperlukannya

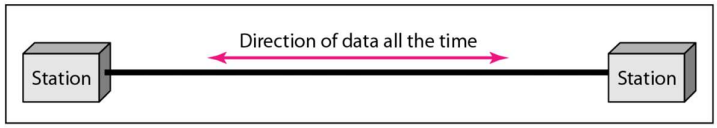
komunikasi dua arah pada saat bersamaan, seluruh kapasitas saluran dapat dimanfaatkan untuk setiap arah.



Gambar 2.8 Komunikasi Data Menggunakan *Half-Duplex*.

3. *Full-Duplex*

Pada mode *full-duplex*, kedua stasiun dapat melakukan transmisi dan menerima transmisi secara bersamaan. Mode *full-duplex* bisa dikatakan seperti sebuah jalan raya dua arah dengan lalu lintas yang mengalir di kedua arah pada saat yang sama. Dalam mode *full-duplex*, sinyal masuk pada satu arah berbagi kapasitas jalur dengan sinyal keluar ke arah lain. Pembagian ini dapat terjadi dalam dua cara, entah dari dua koneksi stasiun yang terhubung harus memiliki dua jalur transmisi fisik secara terpisah, satu jalur untuk melakukan transmisi dan yang lainnya untuk menerima transmisi. Bisa juga dengan membagi kapasistas kanal antara sinyal yang ditransmisikan dan penerima transmisi sinyal. Contoh umum dari komunikasi *full-duplex* adalah jaringan telepon. Ketika dua orang yang berkomunikasi dengan saluran telepon, keduanya dapat berbicara dan mendengarkan pada saat yang bersamaan.



Gambar 2.9 Komunikasi Data Menggunakan *Full-Duplex*.

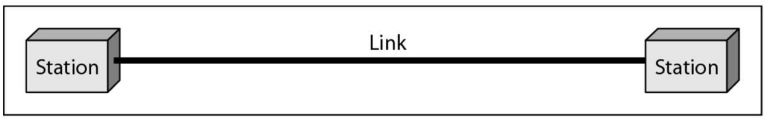
2.3.2 Jenis Koneksi

Sebuah jaringan merupakan dua atau lebih perangkat yang terhubung melalui suatu koneksi. Koneksi sendiri adalah jalur komunikasi yang memindahkan data dari satu perangkat ke perangkat lain. Sebagai gambaran mudahnya, koneksi bisa dibayangkan seperti garis yang terhubung antara dua titik. Agar komunikasi bisa

dilakukan, dua perangkat mesti terhubung dalam jalur yang sama pada saat itu juga. Ada dua contoh dari jenis koneksi, yaitu *point-to-point* dan *multipoint*.

2.3.2.1 *Point-to-Point*

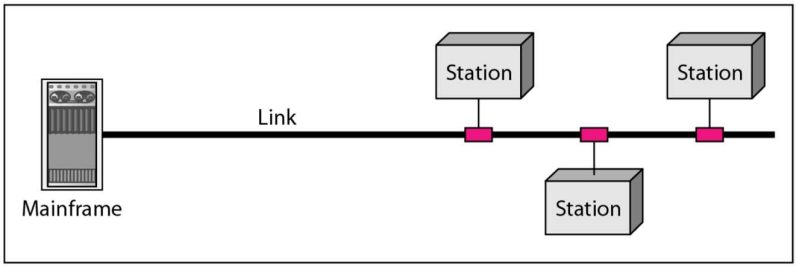
Point-to-point menyediakan jaringan sendiri antara dua perangkat. Semua kapasitas dari jaringan telah diperuntukkan sebagai transmisi antara dua perangkat tersebut. Kebanyakan koneksi *point-to-point* menggunakan kabel untuk menghubungkan kedua titik, akan tetapi ada pilihan lain, yaitu *microwave* atau jaringan satelit juga memungkinkan digunakan. Ketika memindahkan saluran televisi dengan inframerah *remote*, sebenarnya terjadi suatu koneksi *point-to-point* antara remot dan televisi.



Gambar 2.10 Koneksi *Point-to-Point*.

2.3.2.2 *Multipoint*

Multipoint atau biasa disebut dengan koneksi *multidrop* adalah satu jaringan yang mana terdapat dua perangkat atau lebih. Pada *multipoint*, terbaginya kapasistas kanal, baik secara spasial atau temporal. Jika beberapa perangkat bisa menggunakan jaringan secara bersamaan, maka terjadi pembagian koneksi secara spasial. Jika banyak *user* harus bergiliran, maka terjadi koneksi *timeshared*.



Gambar 2.11 Koneksi *Multipoint*.

2.3.3 Protokol TCP/IP

Protokol TCP/IP dikembangkan sebelum adanya model OSI. Oleh karena itu, lapisan dalam TCP/IP berbeda dengan model OSI. Protokol TCP/IP memiliki empat lapisan, yaitu *host-to-network*, *internet, transport,* dan *application*. Namun, ketika

TCP/IP dibandingkan dengan OSI, bisa dikatakan bahwa lapisan *host-to-network* sama dengan kombinasi dari *physical layer* dan *data link layer*. Lapisan *internet* sama dengan *network layer*, dan lapisan *application* bisa disamakan dengan *session layer*, *presentation layer*, dan *application layer*, tetapi lapisan *transport* pada TCP/IP mengurus bagian dari tugas *session layer*. Jadi bisa diasumsikan bahwa protokol TCP/IP terbuat dari lima lapisan, yaitu *physical*, *data link*, *network*, *transport*, dan *application*. Empat lapisan pertama mengurus standar perangkat fisik, antar muka jaringan, *internetworking*, dan melakukan pengiriman yang sesuai dengan empat lapisan pertama dari model OSI. Tiga lapisan paling luar dari model OSI bila dalam protokol TCP/IP dapat terwakilkan oleh lapisan *application*.

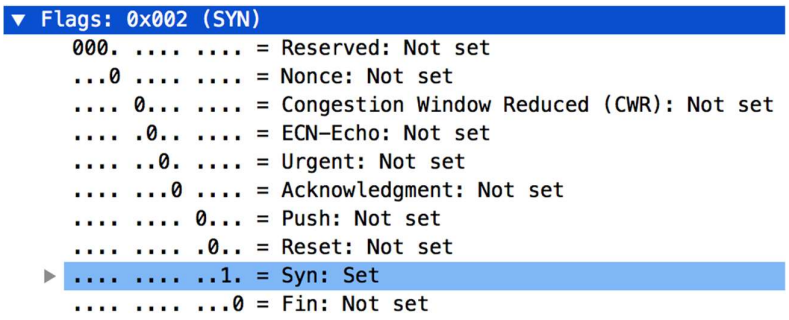
OSI Layers	TCP/IP Suit	Activities
Application	**Application** Telnet, FTP, SMTP, HTTP, DNS, SNMP, *Specific address* etc...	To allow access to network resources
Presentation	**Presentation**	To Translate, encrypt, and compress data
Session	**Session**	To establish, manage, and terminate session
Transport	**Transport** SCTP, TCP, UDP, Sockets and *Ports address*	To Provide reliable process-to-process Message delivery and error recovery
Network	**Network** IP, ARP/RARP, ICMP, IGMP, *Logical address*	To move packets from source to destination; to provide internetworking
Data Link	**Data Link** IEEE 802 Standards, TR, FDDI, PPP, *Physical address*	To organize bits into frames; to provide Hop-to-hop delivery
Physical	**Physical** Medium, Coax, Fiber, 10base, Wireless	To Transmit bits over a medium; to provide Mechanical and electrical specifications

Gambar 2.12 Lapisan Model OSI dan Fungsinya.

Pada bagian *transport layer* di model OSI, TCP/IP memiliki tiga protokol, yaitu *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), dan *Stream Control Transmission Protocol* (SCTP). Pada bagian *network layer* di model OSI, protokol paling utama TCP/IP adalah *Internetworking Protocol* (IP) meskipun ada beberapa protokol lain yang dapat mendukung perpindahan data dalam lapisan ini.

Protokol TCP merupakan protokol yang beroirientasikan koneksi. Berorientasi koneksi pada TCP maksudnya adalah setiap *client-server* yang ingin

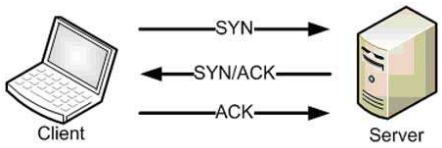
melakukan komunikasi data wajib melakukan suatu persetujuan komunikasi atau biasanya sering disebut dengan *three-way handshake*. *Three-way handshake* sendiri merupakan 3 tahapan *client* dan *server* yang ingin melakukan komunikasi data dengan mengirimkan *flag* TCP dari *client* dan ke *server*. *Flag* TCP dapat dilihat pada Gambar 2.13.



Gambar 2.13 Contoh *Flag* TCP Pada *Wireshark*.

Untuk melakukan *handshake*, *flag* yang dibutuhkan adalah *flag* SYN dan ACK. Berikut adalah *three-way handshake*:

1. *Client* mengirimkan *flag* SYN ke *server*.
2. *Server* mengimkan balasan *flg* SYN dan ACK ke *client*.
3. *Client* mengirimkan balasan ACK ke *server*.



Gambar 2.14 *Three-Way Handshake* Protokol TCP.

2.4 Pemrograman Terstruktur

Pemrograman terstruktur adalah bahasa pemrograman yang mendukung pembuatan program sebagai kumpulan prosedur. Prosedur-prosedur ini dapat saling memanggil dan dipanggil dari manapun dalam program dan dapat mengunakan parameter yang berbeda-beda untuk setiap pemanggilan. Bahasa pemrograman terstruktur adalah pemrograman yang mendukung abstraksi data, pengkodean terstruktur dan kontrol program terstruktur. Sedangkan Prosedur adalah bagian dari program untuk melakukan operasi-operasi yang sudah ditentukan dengan menggunakan parameter tertentu.

Pada pemrograman terstruktur, seorang pembuat program harus memecahkan kode yang besar kedalam sub rutin yang lebih pendek sebagai kesatuan logika yang lebih kecil, sehingga dapat dimengerti secara mudah. Pada pemrograman terstruktur diusahakan untuk menggunakan variabel global sedikit mungkin, dan sebagai gantinya, sub rutin harus menggunakan variabel lokal serta menggunakan argumen (parameter) sebagai referensi data diantaranya.

Hypertext Preprocessor (PHP) merupakan sebuah bahasa pemrograman yang terpasang pada *Hyper Text Markup Language* (HTML) dan lebih sering digunakan pada layanan *web*. Sebagian besar bahasa pemrograman PHP hampir mirip dengan bahasa C, Java dan Perl. Tujuan utama penggunaan bahasa ini adalah untuk memungkinkan perancang *web* dinamis dengan cepat dan terkomunikasi dengan basis data.

2.4.1.1 Basis Data

Basis data adalah kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah atau dimanipulasi menggunakan perangkat lunak untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur data dan juga batasan-batasan pada data yang akan disimpan. Basis data juga merupakan aspek yang sangat penting dalam sistem informasi karena berfungsi sebagai gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat mengorganisasi data, menghidari duplikasi data, menghindari hubungan antar data yang tidak jelas dan juga update yang rumit.

Basis Data Oracle 11g adalah salah satu produk Oracle Corporation yang sampai saat ini telah dipercaya sebagai perangkat lunak *Relational Database Management System* (RDBMS) dengan bahasa pemrograman *Structured Query Language* (SQL) dan *Prosedural Language*/*Structured Query Language* (PL/SQL). RDBMS sendiri merupakan sistem yang dapat memindahkan data ke dalam basis data, menyimpan data, menerima data, sehingga basis data dapat dimanipulasi dengan berbagai macam aplikasi.

2.5 *Web Services*

Penggunaan *web services* pada *World Wide Web* (www) berkembang dengan pesat seiring dengan kebutuhan komunikasi antar aplikasi dan meningkatnya

interoperabilitas. Layanan ini menyediakan sarana komunikasi standar di antara berbagai aplikasi perangkat lunak yang memberikan informasi secara dinamis kepada *user*[4].

Web services adalah sebuah sistem perangkat lunak yang didesain untuk mendukung interaksi yang interoperabilitas antar mesin melalui sebuah jaringan. Walaupun konsep-konsep yang membentuk SOA (*Service Oriented Architecture*) telah ada sebelum *web services* muncul, *web services* memiliki peran penting di dalam SOA. Hal ini dikarenakan *web services* dibangun di atas protokol-protokol yang sudah terkenal dan memiliki *platform* yang *independen*, seperti HTTP, XML (*Extensible Markup Language*), UDDI (*Universal Discovery Description Integration*), dan WSDL (*Web Services Description Language*)[5].

2.6 *Web Server*

Web server merupakan perangkat lunak yang memberikan layanan data, berfungsi menerima permintaan HTTP atau HTTPS (*HyperText Transfer Protocol Secure*) dari *client* yang dikenal dengan *web browser* dan mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen HTML, konsep *web server* antara lain:

a. *Web server* merupakan mesin aplikasi atau perangkat lunak yang beroperasi dalam mendistribusikan halaman *web* ke *user*, tentu saja sesuai dengan permintaan *user*.
b. Hubungan antara *web server* dan *browser* merupakan gabungan atau jaringan komputer yang berada di seluruh dunia. Setelah terhubung secara fisik, protokol TCP/IP yang memungkinkan semua komputer dapat berkomunikasi antar satu dengan lainnya. Pada saat aplikasi *browser* meminta data halaman *web* ke *server* maka instruksi permintaan data oleh *browser* tersebut dikemas dalam TCP yang merupakan protokol pengiriman dan dikirim ke alamat yang merupakan protokol berikutnya yaitu HTTP. Data yang di-*parsing* dari *browser* ke *web server* disebut sebagai HTTP *request* yang meminta halaman *web* dan kemudian *web server* akan mencari data HTML yang dibutuhkan dan dikemas dalam protokol TCP kemudian dikirim kembali ke *browser*. Data yang dikirim dari *server* ke *browser* disebut sebagai HTTP *response*. Jika data yang diminta

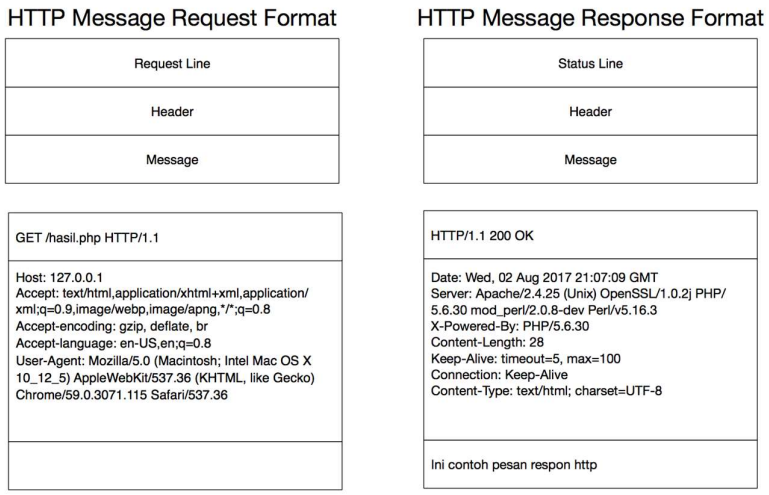
oleh *browser* tidak ditemukan pada *web server* maka akan menampilkan *error* pada halaman *web*, yaitu contohnya *Error: 404 Page Not Found*[6].

2.7 *HyperText Transfer Protocol*

HyperText Transfer Protocol (HTTP) merupakan protokol yang digunakan oleh *web server, web services* dan perangkat yang mampu terhubung ke Internet agar dapat saling berkomunikasi satu sama lain. Protokol ini berada pada tingkat aplikasi untuk sistem informasi terdistribusi, terkumpul, dan dengan berbagai macam media. HTTP telah digunakan pada awal mula *World Wide Web* (www) sejak tahun 1990-an. Versi pertama HTTP lebih dikenal dengan HTTP/0.9 yang merupakan protokol sederhana untuk mengirimkan data mentah melalui Internet. Setelah itu, munculah HTTP/1.0 yang merupakan protokol yang lebih baik, protokol ini mengizinkan sebuah pengiriman pesan dengan format MIME (*Multipurpose Internet Mail Extension*), berisikan elemen-elemen informasi mengenai data yang dikirimkan dan data yang diubah pada *request/response*[7]. Akan tetapi, HTTP/1.0 menggunakan koneksi terpisah untuk setiap pesan yang dikirimkan. Oleh karena itu, versi HTTP/1.1 dapat menggunakan koneksi yang sama untuk melakukan pengiriman.

Protokol HTTP adalah protokol yang menganut *request/response*. *User* mengirimkan sebuah *request* (permintaan) ke dalam *server* dalam bentuk *request method* (metode permintaan), URI (*Uniform Resurce Identifier*), dan versi protokol.

Dalam HTTP, selain berguna sebagai jembatan komunikasi antara beberapa perangkat, HTTP dapat digunakan untuk melakukan penerimaan dokumen dan perbaruan dokumen melalui URI dengan menggunakan *request methods*, yaitu *GET method* dan *POST method*. Kedua *methods* tersebut merupakan sebuah layanan yang sudah diberikan oleh HTTP.



Gambar 2.15 Format HTTP *Response* dan HTTP *Request*.

2.8 *Uniform Resource Identifier*

Uniform Resource Identifier (URI) merupakan sebuah identifikasi yang terdiri dari urutan karakter yang tepat dengan aturan tertentu yang digunakan untuk mengakses suatu *resource* (sumber) melalui sebuah jaringan dengan menggunakan protokol tertentu. URI merepresentasikan entitas dari suatu *resource* yang memiliki format protokol://host/path atau contohnya adalah seperti http://facebook.com/login. HTTP menunjukkan protokol yang digunakan, facebook.com menunjukkan host dari *server*, dan login menunjukkan lokasi berkas yang diakses.

Selain itu, dalam formatnya, URI memiliki yang namanya *query*. *Query* digunakan untuk mengirimkan informasi dari *server* ke *client* atau sebaliknya. *Query* yang umum digunakan ketika melakukan autentikasi pada halaman *login*. Ketika *user* melakukan *input* informasi dari *client*, maka yang dikirimkan ke *server* adalah *query* dari URI. Format dari *query* dimulai dengan menggunakan tanda "?", dan bisa dilihat seperti http://facebook.com/login?username=xxx&password=xxx. Bila dilihat dari contoh, *query* mengirimkan informasi *username* dan *password* ke *server*. Oleh karena itu, *query* bisa digunakan untuk mengirimkan suatu informasi dari *server* ke *client* atau sebaliknya.

2.9 *Dynamic Domain Name System*

Dynamic Domain Name System (DDNS) adalah sistem yang memungkinkan untuk memperbarui IP *Public* dinamis menjadi sebuah nama. Berbeda dengan IP *Public* statis yang tidak akan berubah-berubah. Bila seseorang berlangganan layanan *Internet* ke *Internet Service Provide* (ISP), biasanya provider akan memberikan IP *Public* dinamis kepada pengguna *Internet*-nya. ISP sendiri sebenarnya menyediakan IP *Public* statis kepada pengguna layanan *Internet*-nya, akan tetapi hal tersebut membutuhkan biaya lebih. Oleh karena itu penggunaan DDNS bisa menjadi alternatif bila seseorang yang menggunakan IP *Public* dinamis ingin mengubah IP yang digunakan menjadi sebuah nama.

2.10 *JavaScript Object Notation*

JavaScript Object Notation (JSON) adalah sebuah format teks yang memfasilitasi pertukaran struktur data dengan semua bahasa pemrograman[8]. JSON sendiri merupakan sintaks yang terdiri dari tanda kurung siku, tanda kurung kurawal, titik dua, dan koma yang berguna untuk banyak konteks, profil, dan aplikasi. JSON bukan hanya sekadar menawarkan representasi angka atau urutan angka yang umum digunakan manusia. Semua bahasa pemrograman tahu betul bagaimana memahami urutan digit meskipun bahasa pemrograman itu sendiri tidak memahami representasi dari kontennya. Hal itu memungkinan untuk melakukan pertukaran data. Berikut adalah mengenai pertukaran data menggunakan JSON:

a. Cakupan

   JSON adalah format pertukaran data yang ringan, berbasiskan teks, dan memiliki bahasa pertukaran data independen. Asalnya dari bahasa pemrograman ECMAScript, tetapi bahasa pemrograman yang independen. JSON mendefinisikan aturan-aturan struktur set ringan untuk representasi data struktur portabel.

b. Kesesuaian

   Kesesuaian teks JSOn adalah urutan kode Unicode yang benar-benar sesuai dengan tata bahasa JSON.

c. Acuan Normatif

Dokumen yang dirujuk sangat diperlukan untuk penerapan JSON. Berikut adalah salah satu contoh rujukan yaitu *ISO/IEC 10646:2012, Information Technology - Universal Coded Character Set (UCS)*.

d. Teks JSON

Teks JSON adalah urutan token yang terbentuk dari kode Unicode yang sesuai dengan nilai tata bahasa JSON. Kumpulan token termasuk enam struktural token, *numbers,* dan tiga nama token literal. Berikut adalah enam struktural token:

1. [U+005B Kurung Siku Buka.
2. { U+007B Kurung Kurawal Buka.
3.] U+005D Kurung Siku Tutup.
4. } U+007D Kurung Kurawal Tutup.
5. : U+003A Titik Dua.
6. ,U+002C Koma.

Berikut adalah tiga nama token literal:

1. true U+0074 U+0072 U+0075 U+0065
2. false U+0066 U+0061 U+006c U+0073 U+0065
3. null U+006E U+0075 U+006C U+006C

e. Nilai JSON

Nilai JSON dapat berupa sebuah objek, array, nomor, *string*, *true*, *false*, atau *null*.

```
[{
    "Nomor": "1",
    "Nama": "Bam Frans",
    "Tempat Lahir": "Bandung",
    "Tanggal Lahir": "07-03-1992"
}]
```

Gambar 2.16 Contoh Format Data JSON.

2.11 *Wireshark*

Wireshark adalah perangkat lunak yang mampu melakukan analisis paket pada jaringan. *Wireshark* menangkap paket-paket yang ada pada suatu jaringan dan menampilkan paket-paket data tersebut selengkap mungkin.