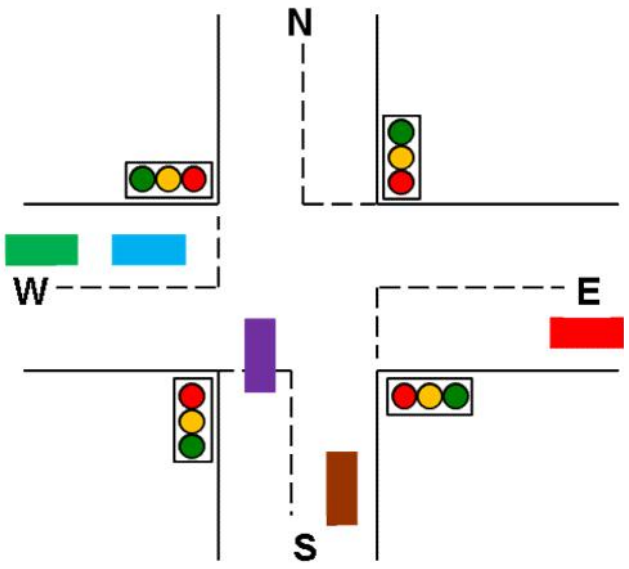
BAB II
TEORI PENUNJANG

2.1 Prinsip Kerja Lampu Lalu Lintas 4 jalur

Prinsip kerja lampu lalu lintas dibangun berdasarkan banyaknya arah di suatu persimpangan dan waktu yang dibutuhkan untuk mengatur arus kendaraan di persimpangan tersebut. Semakin banyak arah yang berlaku di suatu persimpangan maka, semakin banyak lampu lalu lintas yang digunakan. Banyaknya lampu akan membuat algoritma pemrogramannya semakin rumit. Adapun pengaruh waktu saat mengatur arus kendaraan terhadap kinerja sistem lampu lalu lintas. Semakin lama waktu yang dibutuhkan maka, semakin banyak antrian kendaraan. Perlu diketahui bahwa jumlah antrian tidak boleh terlalu panjang agar tidak memicu kemacetan di persimpangan lain.

Pada kenyataannya, lampu lalu lintas digunakan pada persimpangan berarus kendaraan besar. Jenis persimpangan ini biasanya berupa persimpangan 4 jalur atau lebih dikenal dengan simpang 4.



Gambar 2.1 Persimpangan 4 jalur

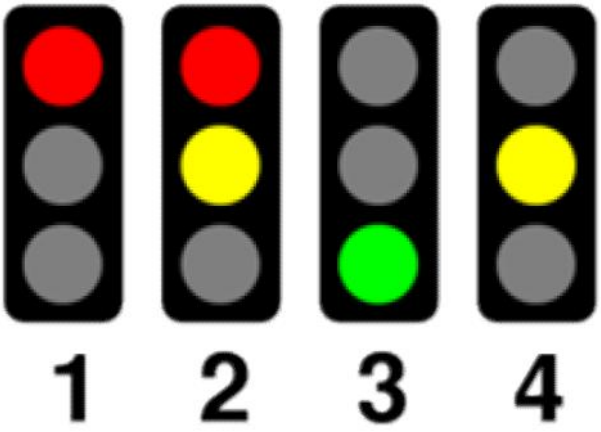
Adapun prinsip yang berlaku pada sistem lampu lalu lintas ini ialah sebagai berikut:

1. Lampu hijau di salah satu jalur menyala artinya kendaraan pada jalur tersebut boleh berjalan ke tiga jalur lainnnya.
2. Lampu merah di jalur manapun menyala, maka kendaraan pada jalur tersebut harus berhenti dan hanya boleh berjalan ke arah jalur kirinya (UU Lalu Lintas No.14 tahun 1992) .

Kinerja suatu sistem lampu lalu lintas ditentukan oleh waktu yang digunakan untuk mengatur arus kendaraan. Terdapat 2 pengoperasian yang digunakan pada pengontrolan lampu lalu lintas yaitu:

1. Fixed Time Traffic Signal adalah pengoperasian dimana waktu yang digunakan tidak mengalami perubahan.
2. Actuated Traffic Signal adalah pengoperasian dimana waktu yang digunakan mengalami perubahan sesuai keadaan persimpangan.

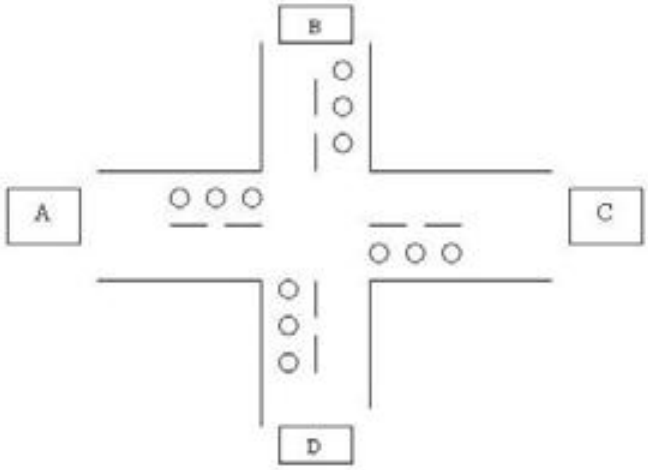
Perlu diperhatikan bahwa waktu tersebut ditentukan berdasarkan hasil analisa yang telah dilakukan sebelum mendirikan suatu lampu lalu lintas. Hasil analisa ini dapat berupa lamanya lampu merah, lamanya lampu hijau, dan giliran pergantian lampu antara lampu lalu lintas dengan lampu lalu lintas lainnya di suatu persimpangan. Lampu secara bergiliran akan berganti sesuai dengan waktu tersebut. Adapun urutan giliran lampu lalu lintas yang telah ditetapkan berdasarkan ketentuan internasional dapat dilihat pada gambar dibawah ini:



Gambar 2.2 Urutan Pergantian Lampu Lalu Lintas

2.2 Penerapan Algoritma Welch-Powell pada Operasi Persimpangan 4 Jalur

Algoritma *Welch-Powell* merupakan salah satu teknik dari pewarnaan graf untuk mengatasi masalah yang muncul dari penyusunan jadwal. Algoritma *Welch-Powell* dapat diterapkan untuk menentukan pola lampu lalu lintas dengan jumlah fase minimal, dan pada setiap fase tidak ada perjalanan yang saling melintas. Perjalanan yang diperbolehkan adalah : A ke B, A ke C, A ke D, B ke C, B ke D, B ke A, C ke D, C ke A, C ke B, D ke A, D ke B, dan D ke C.

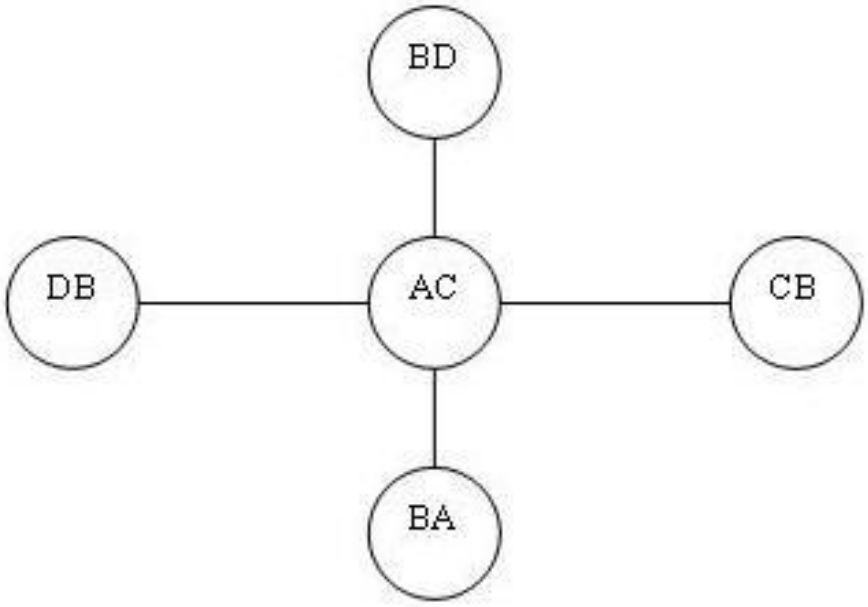


Gambar 2.3 Persimpangan Empat Jalur

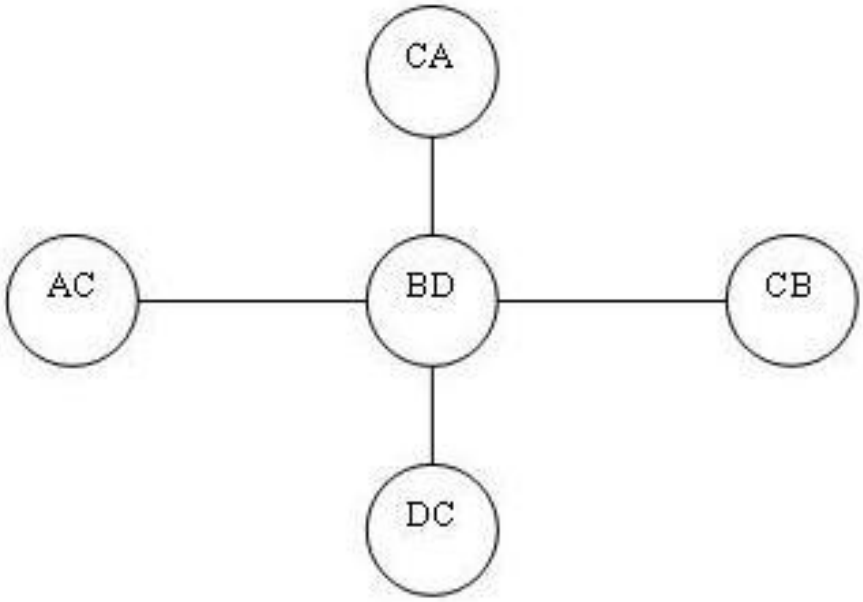
Langkah-langkah penyelesaiannya :

1. Tentukan simpul dari perjalanan yang diperbolehkan (untuk peletakan simpulnya bebas).
2. Tentukan ruas yang menghubungkan 2 simpul yang menyatakan 2 perjalanan yang saling melintas.

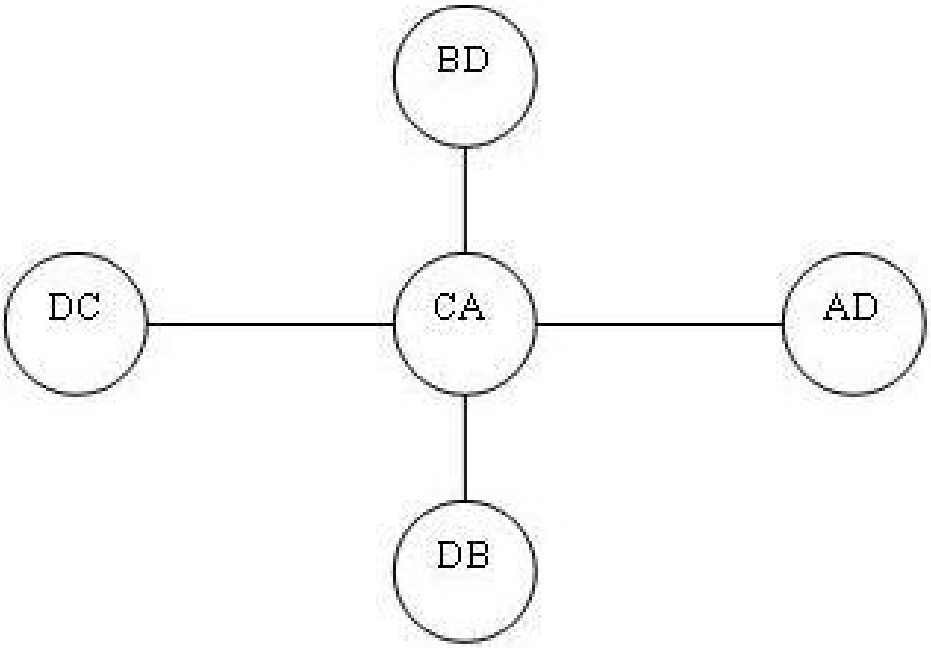
Simpul-simpul yang diperbolehkan jalan pada semua kondisi adalah AB, BC, CD, dan DA. Simpul yang menyatakan dua perjalanan saling melintas adalah :



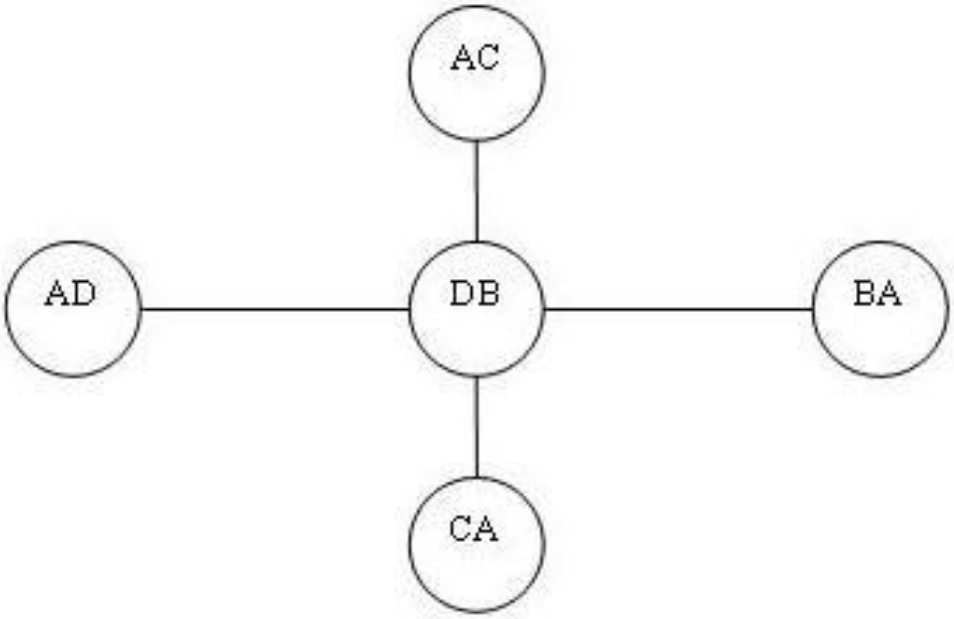
Gambar 2.4 Simpul yang saling melintas dengan AC



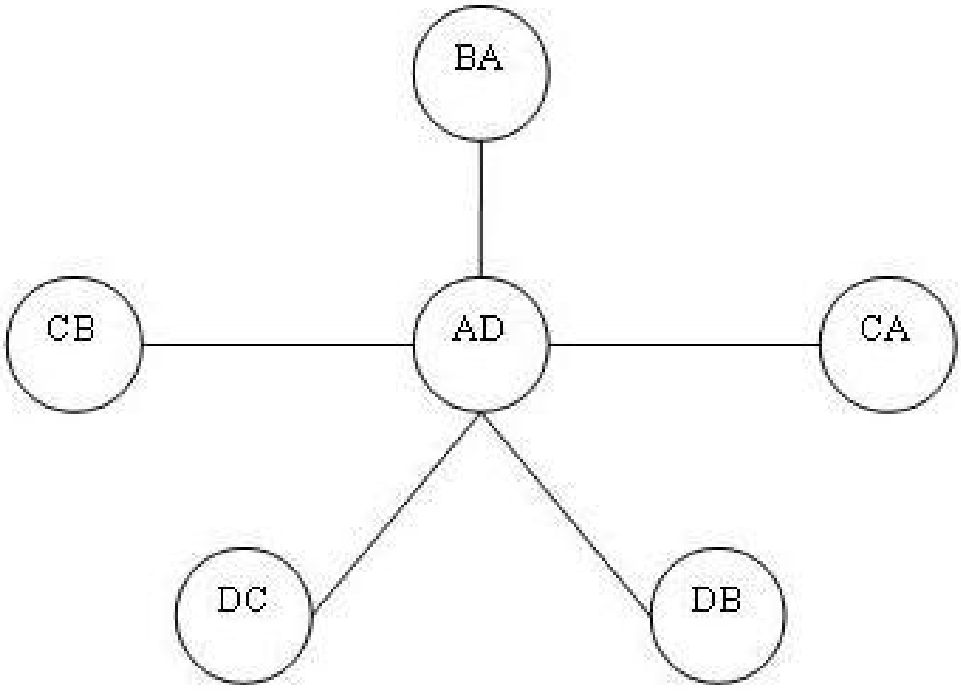
Gambar 2.5 Simpul yang saling melintas dengan BD



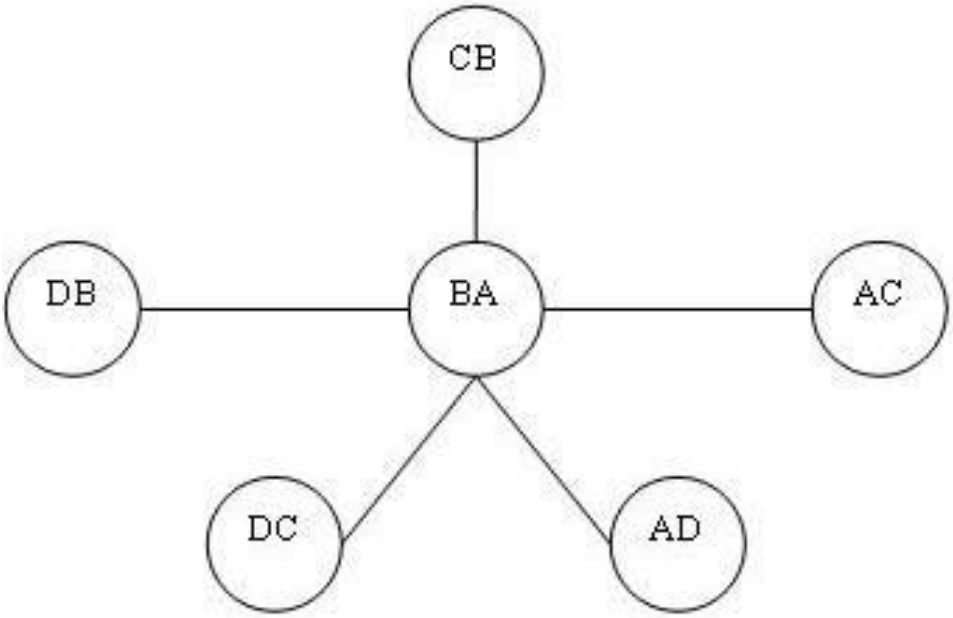
Gambar 2.6 Simpul yang saling melintas dengan CA



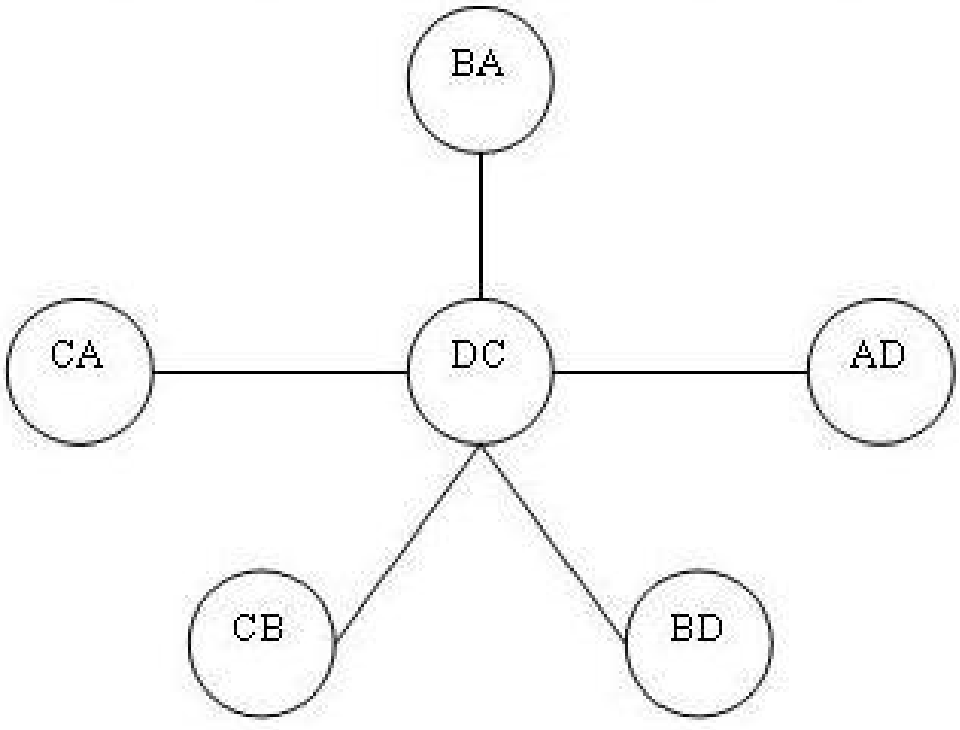
Gambar 2.7 Simpul yang saling melintas dengan DB



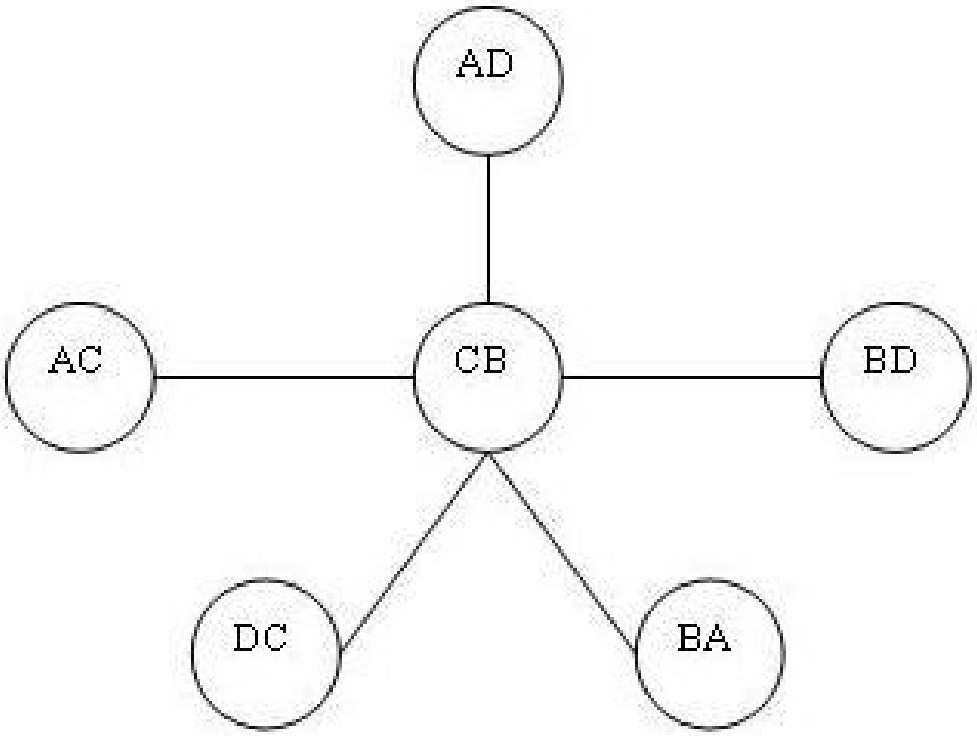
Gambar 2.8 Simpul yang saling melintas dengan AD



Gambar 2.9 Simpul yang saling melintas dengan BA



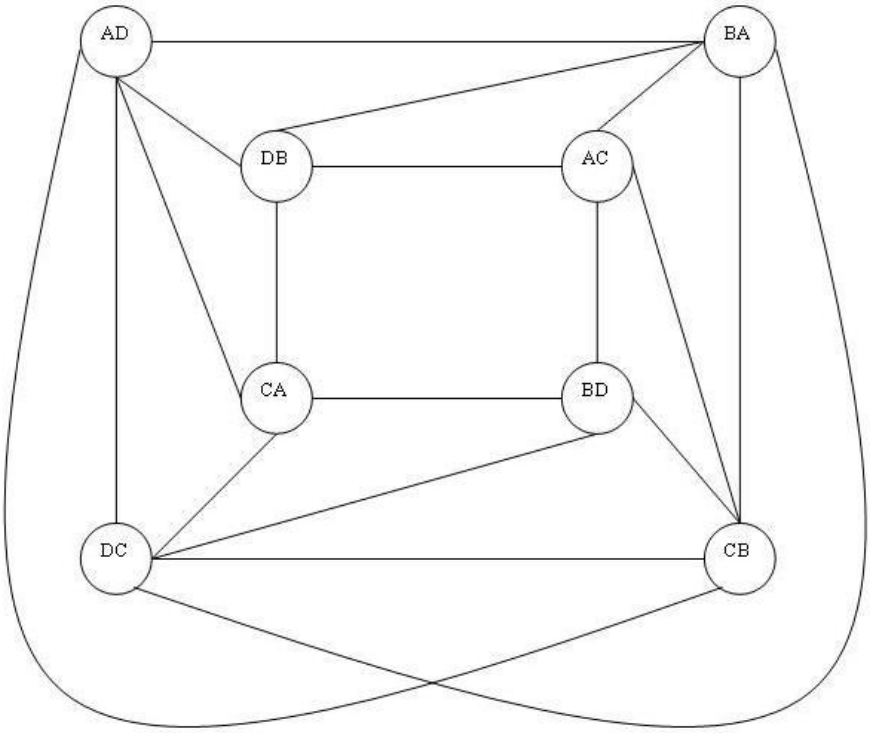
Gambar 2.10 Simpul yang saling melintas dengan DC



Gambar 2.11 Simpul yang saling melintas dengan CB

Maksud dari gambar-gambar simpul tersebut adalah apabila suatu simpul dihubungkan dengan simpul lain (yang saling melintas) menandakan bahwa satu simpul diartikan sebagai perintah jalan dan simpul lainnya diartikan harus menunggu atau berhenti.

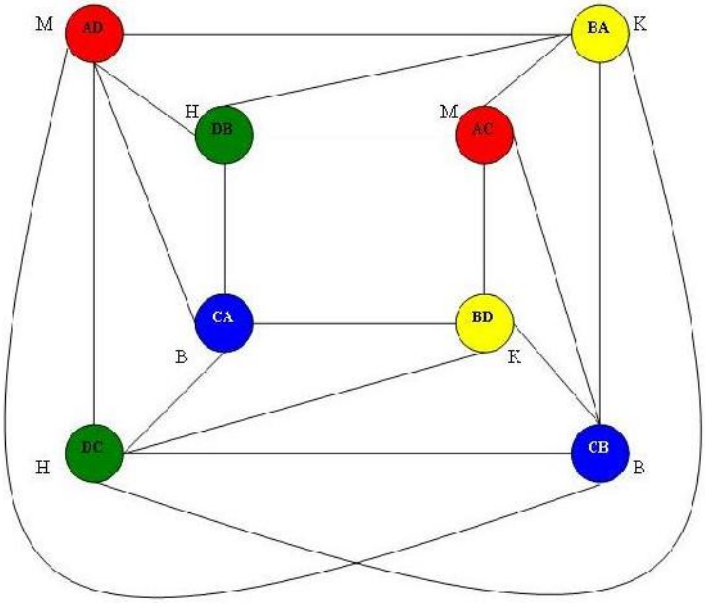
Setelah menentukan ruas yang menghubungkan 2 simpul yang menyatakan 2 perjalanan saling melintas. Diketahui bahwa 8 simpul saling bersinggungan saat sama-sama melintas. Langkah selanjutnya menyatukan simpul dan ruas menjadi satu-kesatuan yang utuh.



Gambar 2.12 Simpul Secara Keseluruhan

Kemudian diurutkan simpul yang memiliki derajat simpul terbesar hingga yang memiliki derajat simpul terkecil Adapun urutannya yaitu, d(AD) = 5, d(BA) = 5, d(CB) = 5, d(DC) = 5, d(AC) = 4, d(BD) = 4, d(CA) = 4, d(DB) = 4.

Beri warna pada setiap simpul dengan warna baru, bila simpul berdampingan maka berilah warna lain, bila simpul tidak berdampingan maka berilah warna yang sama.



Gambar 2.13 Simpul Keseluruhan yang Telah Diberi Warna

Langkah-langkah untuk mewarnai setiap simpul adalah sebgai berikut:

1. Beri warna pada setiap simpul secara berurutan dari yang memiliki derajat simpul terbesar sampai dengan derajat simpul terkecil. Urutannya adalah : AD, BA, CB, DC, AC, BD, CA, DB. Ambil warna pertama misalnya merah.
2. Beri warna merah simpul AD (karena AD adalah simpul urutan yang pertama). Kemudian cari simpul yang *tidak berdampingan* dengan simpul AD, beri warna yang sama (merah). Kita berikan warna yang sama pada simpul AC dengan warna simpul AD yaitu merah, karena simpul AC tidak berdampingan dengan simpul AD.
3. Sehingga didapat urutan simpul yang belum di beri warna sebagai berikut : BA, CB, DC, BD, CA, DB.
4. Ambil warna kedua, misalnya kuning. Warnai simpul BA (karena simpul BA sekarang ada diurutan yang pertama). Kemudian cari simpul yang tidak berdampingan dengan simpul BA, beri warna yang sama (kuning). Kita berikan warna yang sama pada simpul BD dengan warna simpul BA yaitu kuning, karena simpul BD tidak berdampingan dengan simpul BA. Sehingga didapat urutan simpul yang belum di beri warna yaitu, CB, DC, CA, DB.

5. Ambil warna ketiga, misalnya biru. Warnai simpul CB (karena simpul CB sekarang ada diurutan yang pertama). Kemudian cari simpul yang tidak berdampingan dengan simpul CB, beri warna yang sama (biru). Kita berikan warna yang sama pada simpul CA dengan warna simpul CB yaitu biru, karena simpul CA tidak berdampingan dengan simpul CB. Sehingga didapat urutan simpul yang belum di beri yaitu DC, DB.
6. Ambil warna keempat, misalnya Hijau. Warnai simpul DC dan DB. Pewarnaan simpul telah selesai. Graf merupakan graf berwarna 4. Jadi bilangan kromatisnya adalah K(G) = 4.

Simpul AB, BC, CD, dan DA tidak dihubungkan oleh suatu ruas jadi untuk simpul tersebut tidak pernah melintas perjalanan-perjalanan lain dan simpul tersebut selalu berlaku lampu hijau. Berdasarkan langkah-langkah algoritma *Welch-Powell* diatas, maka diperoleh hasil sebagai berikut :

1. Fase 1
   Merah : (BD, CA, DB, BA, CB, DC)
   Hijau : (AD, AC, AB, CD, BC, DA)
2. Fase 2
   Merah : (AC, CA, DB, AD, CB, DC)
   Hijau : (BA, BD, AB, CD, BC, DA)
3. Fase 3
   Merah : (AC, BD, DB, AD, BA, DC)
   Hijau : (CB, CA, AB, CD, BC, DA)
4. Fase 4
   Merah : (AC, BD, CA, AD, BA, CB)
   Hijau : (DC, DB, AB, CD, BC, DA)

2.3 Embedded Web Server

Secara harfiah, embedded web server adalah suatu sistem yang sengaja ditanamkan ke suatu jaringan komputer agar dapat difungsikan sebagai web server. Ide tentang embedded web server ini berasal dari interaksi antara nilai kebutuhan terhadap kinerja web server dengan nilai

biaya yang perlu dikeluarkan. Jika web server memiliki prosesor yang cepat dan RAM yang sangat besar, maka prosesor pada versi embedded web server lebih lambat dan memiliki RAM yang lebih kecil. Namun, biaya yang dikeluarkan untuk embedded web server jauh lebih murah daripada biaya untuk web server.

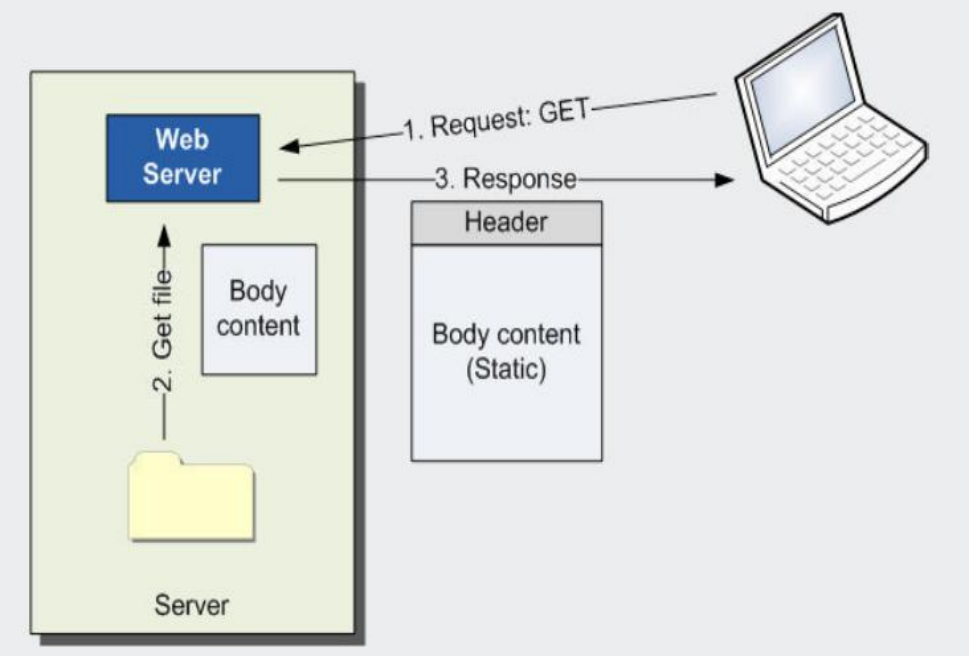


Gambar 2.14 Rainer SM150C8-2.2 SATA35NR 2x 4GB Tower Server

Awalnya embedded web server ini merupakan suatu bentuk sistem yang berbeda. Dikarenakan, karakteristik yang sama dan dapat menopang fungsi server sebagai web server, maka terciptalah embedded system yang berfungsi sebagai web server.

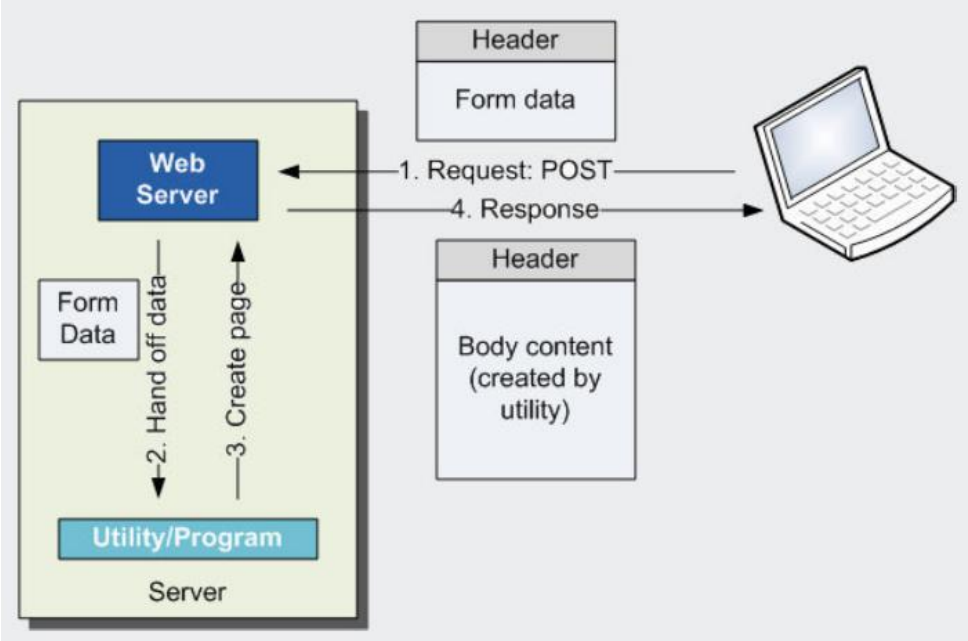
Perlu diketahui bahwa sebuah *web server* memiliki satu peran yaitu menjalankan protokol HTTP. Protokol ini memungkinkan browser untuk mengirimkan request kepada *server* yang akan menerima. Operasi request yang dimaksud ialah operasi get dan operasi post. Operasi get digunakan untuk men-download "sumber daya" (biasanya file) yang terletak di lokasi

yang ditentukan atau lebih dikenal sebagai URL (*Uniform Resource Locator*). Ketika seseorang mengklik link untuk pergi ke suatu halaman pada web browser, HTTP mengirimkan permintaan GET ke server dan meminta halaman HTML yang terletak di URL yang diklik.



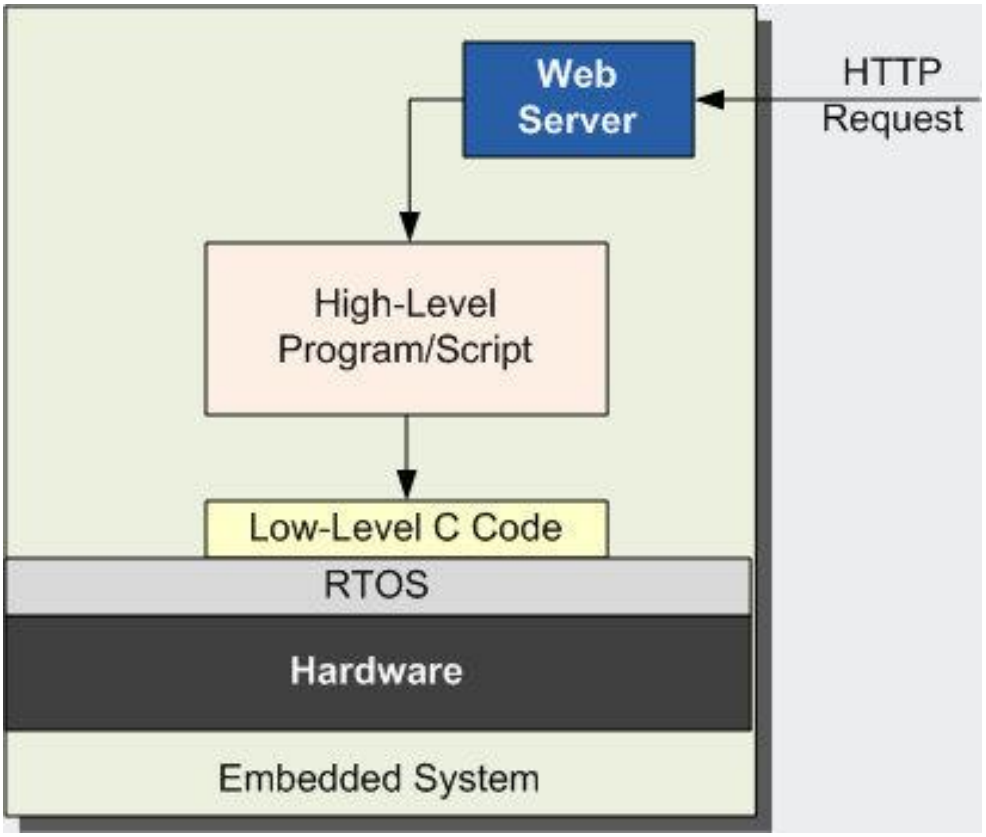
Gambar 2.15 Deskripsi Permintaan GET

Operasi Post digunakan untuk mengirimkan data dari web browser kembali ke server. Ini adalah permintaan yang dihasilkan ketika seseorang menekan tombol "Kirim" pada formulir yang diisi.



Gambar 2.16 Deskripsi Permintaan POST

Secara khusus, aplikasi embedded web pada umumnya harus mampu mempengaruhi operasi perangkat keras. Sebuah *web server* tidak memiliki kemampuan seperti itu. Sehingga segala sesuatu di luar pengolahan HTTP harus dinyatakan sebagai bagian dari pengembangan sistem. Kode tingkat rendah yang akan berinteraksi langsung dengan perangkat keras akan ditulis dalam pemrograman berbahasa C. Dikarenakan rutin ini begitu khusus, mereka akan perlu ditulis selama pengembangan tidak peduli bagaimana konektivitas web diimplementasikan. Masalahnya adalah web server tidak memiliki cara untuk mengakses rutin tersebut. Oleh sebab itu, harus ada cara untuk menghubungkan keduanya yaitu dengan membuat sebuah program yang dapat mengambil permintaan HTTP yang kemudian memanggil kode C.



Gambar 2.17 Deskripsi Embedded Web Server