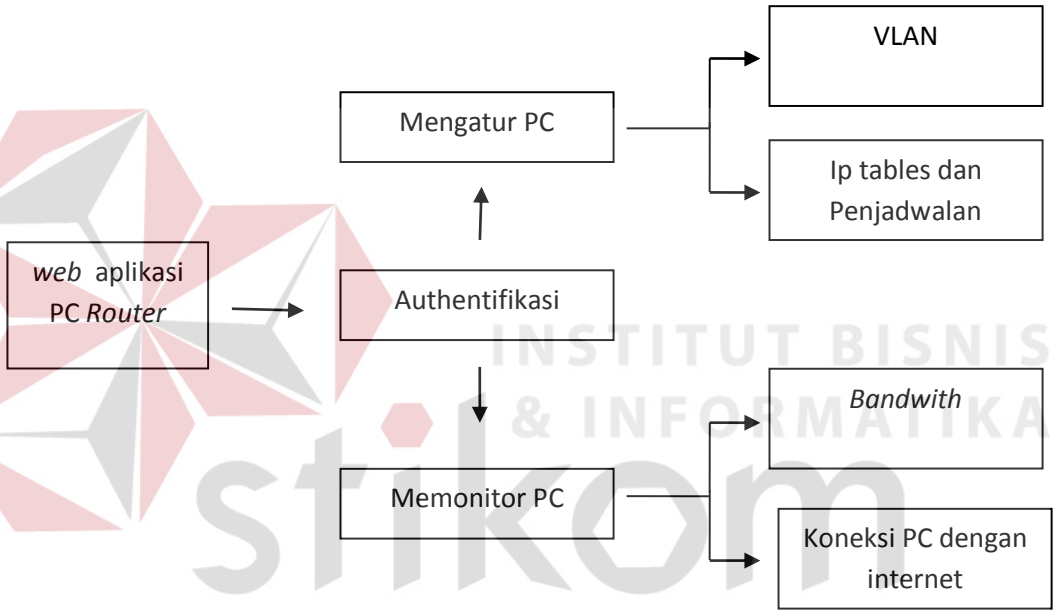
BAB III

METODE PENELITIAN

3.1 Perancangan Sistem dan Blok Diagram Sistem

Perancangan sistem dapat dijelaskan dengan lebih baik melalui blok diagram seperti yang terlihat pada Gambar 3.1 dibawah ini :



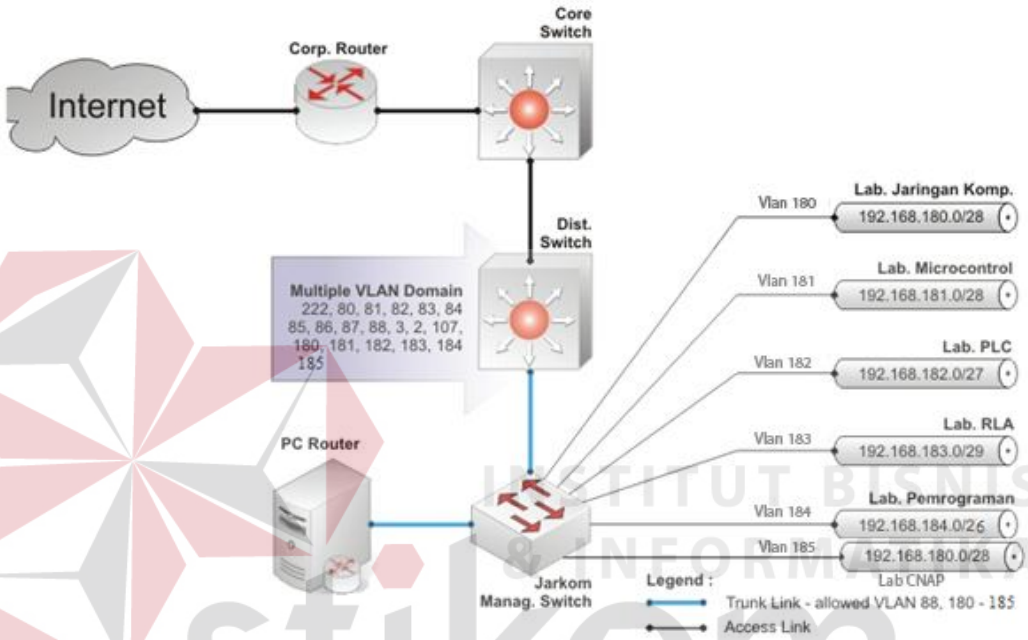
Gambar 3.1 Blok diagram

Gambar 3.1 merupakan blok diagram keseluruhan aplikasi yang akan dibuat. *Web* aplikasi diatas akan tertanam di PC *Router*. *User* yang akan mengakses PC *Router* harus memasukan authentifikasi, dimana orang yang berwewenang yang dapat masuk kedalam aplikasi yang dibuat.

Aplikasi yang dibuat terbagi atas 2 bagian utama, bagian pertama adalah untuk mengatur PC *Router* dan bagian kedua adalah untuk memonitor PC *Router*. Mengatur yang dimaksud adalah mengatur pembuatan VLAN, mengatur IP tables serta penjadwalanya. Sedangkan memonitor yang dimaksud adalah memonitor

bandwith tiap VLAN dan memonitor koneksi PC tiap laboratorium, apakah terhubung dengan PC *Router* atau tidak.

PC *Router* Ubuntu untuk pembuatan aplikasi ini diujicobakan pada laboratorium sistem komputer di lantai 8 gedung Stikom Surabaya dengan topologi sebagai berikut:



Gambar 3.2 Topologi jaringan S1 Sistem Komputer

PC *Router* berfungsi untuk mengoneksikan ke enam laboratorium yaitu laboratorium jaringan dengan VLAN 180, laboratorium mikrokontroler dengan VLAN 181, laboratorium PLC (Programmable Logic Controller) dengan VLAN 182, laboratorium RLA dengan VLAN 183, laboratorium digital dengan VLAN 184 dan dengan laboratorium CNAP dengan VLAN 185. Untuk koneksi internet PC *Router* melakukan proses NAT (*Network* Address Translation) dengan VLAN 88, yaitu VLAN yang telah terkoneksi dengan internet.

Aplikasi ini adalah aplikasi berbasis *web*, oleh karena itu penulis menggunakan PHP dalam implementasinya. Hal yang terpenting untuk

pembuatan aplikasi ini adalah pengaksesan shell. Dimana Setiap sintak yang berjalan pada terminal dapat dijalankan pada PHP. Untuk pengaksesan shell pada PHP dapat dilakukan dengan sintak php_shell (“perintah”). Contoh: $contoh= shell_exec(ifconfig), hal ini berarti isi variabel contoh adalah hasil dari ifconfig terminal.

Hal yang perlu diperhatikan lainya adalah otoritas *user*. *User* yang membuka halaman *web*, menempati kedudukan sebagai var-www. Secara default *user* yang dapat mengakses sistem adalah root. Oleh karena itu, sebelum dijalankan penulis memberi otoritas pada var-www untuk mempunyai wewenang menjalankan perintah sistem.

3.2 Perancangan Pengaturan pada PC *Router*

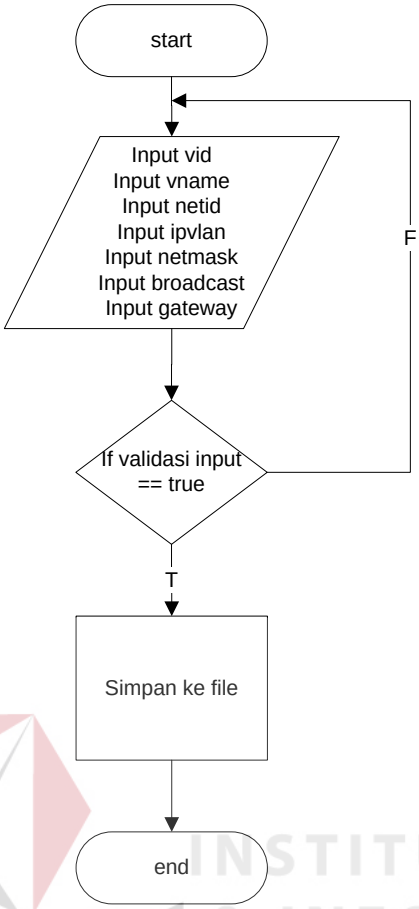
Perancangan pengaturan PC *Router* dalam aplikasi ini terdiri dari 3 bagian utama, yaitu pengaturan VLAN, pengaturan iptables dan pengaturan penjadwalan iptables.

3.2.1 Pengaturan VLAN

Pengaturan VLAN yang dimaksud adalah membuat VLAN, menghapus VLAN, melihat VLAN yang dibuat dan menggonfirmasi VLAN yang dibuat kedalam PC *Router*.

A. Pembuatan VLAN

Perancangan pembuatan VLAN dapat dijelaskan dengan *flowchart* sebagai berikut:



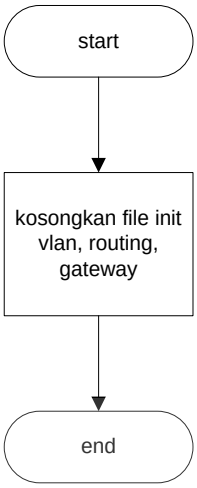
Gambar 3.3 *flowchart* pembuatan VLAN

Proses pembuatan VLAN merupakan proses awal dari penggunaan aplikasi ini. form ini akan menangkap inputan *user* dan menyimpanya kedalam file tertentu. Data yang harus dimasukan oleh *user* adalah VLAN id (nomor VLAN), nama VLAN, *network* id, ip VLAN, netmask VLAN, *broadcast*, dan gateway dari PC *Router*.

Pada dasarnya konsep pembuatan VLAN ini adalah membuat file interfaces sendiri. Input dari *user* membentuk 4 file yang berbeda, file pertama adalah init, sebagai inisialisasi di interface, file kedua adalah berisi konfigurasi VLAN, file ketiga adalah berisi konfigurasi *routing*, dan file keempat adalah berisi konfigurasi gateway dan *path* file iptables. Untuk mengkonfirmasi perubahan VLAN *user* harus membuka halaman apply-VLAN.

B. Penghapusan VLAN

Perancangan pembuatan VLAN dapat dijelaskan dengan *flowchart* sebagai berikut:

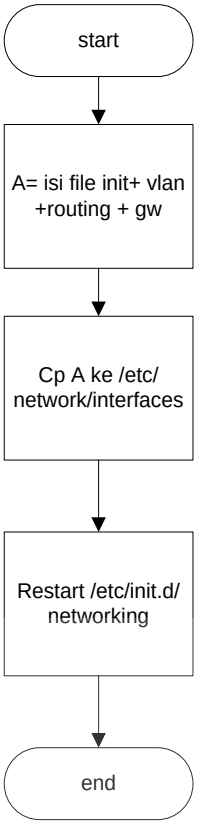


Gambar 3.4 *flowchart* penghapusan VLAN

Konsep penghapusan VLAN dilakukan dengan menghapus keempat file yang dibuat yaitu file konfigurasi VLAN, file konfigurasi *routing*, file konfigurasi gateway kecuali file init. File init berisi *loopback* dan auto eth0, sehingga file init harus selalu ada pada file interfaces.

C. Konfirmasi VLAN

Perancangan proses konfirmasi VLAN dapat dijelaskan dengan *flowchart* berikut:

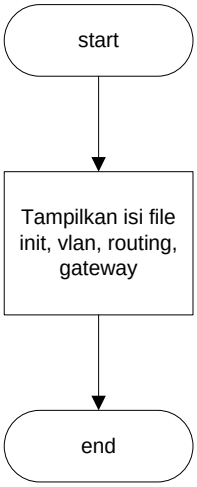


Gambar 3.5 *flowchart* apply-VLAN

Konfirmasi VLAN dimulai dengan menangkap isi dari file init, konfigurasi VLAN, konfigurasi *routing* dan konfigurasi gateway dari proses membuat VLAN. Isi ke empat file tersebut dimasukan kedalam satu file misalnya copy-interfaces. Kemudian *copy copy-interfaces* kedalam file /etc/*network*/interfaces. Setelah itu restart *service* dari *network*, *service network* berada pada /etc/init.d/*network*ing.

D. Menampilkan VLAN

Perancangan proses menampilkan VLAN dapat dijelaskan dengan *flowchart* berikut:



Gambar 3.6 *flowchart* melihat isi VLAN

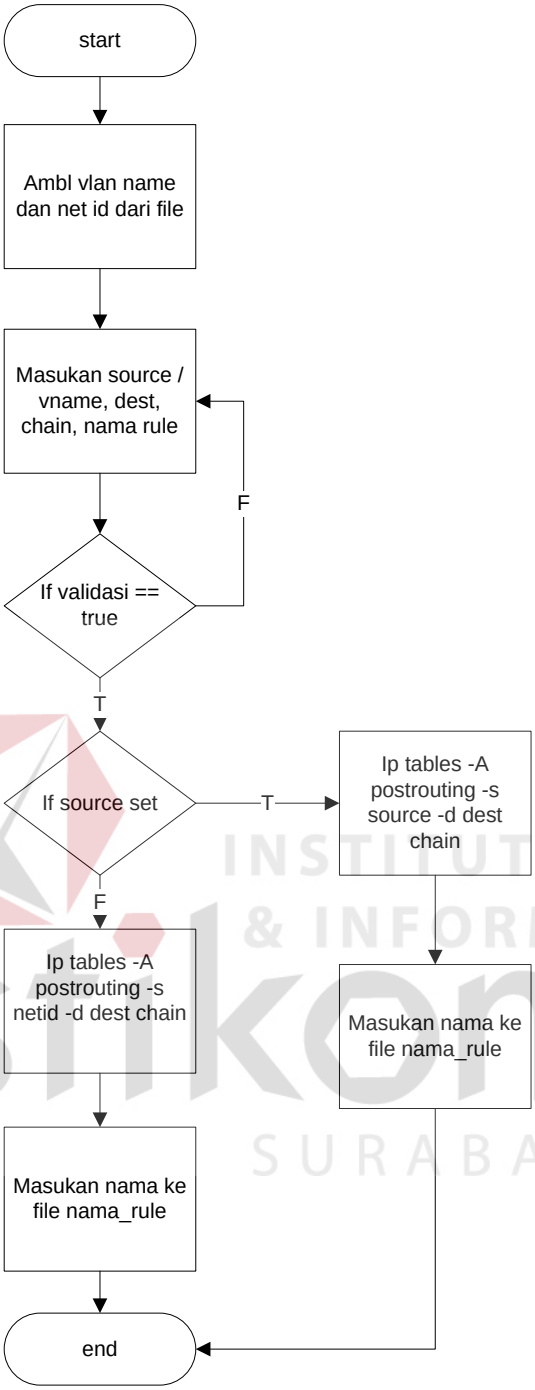
Perancangan menampilkan VLAN dilakukan dengan menampilkan 4 file yaitu konfigurasi init, konfigurasi VLAN, konfigurasi *routing* dan kofigurasi gateway. Keempat file tersebut ditampilkan ke page *list*-perubahan VLAN.

3.2.2 Pengaturan Iptables

perancangan ip tables meliputi 3 hal, yaitu menambah iptables, menghapus iptables, dan melihat isi dari iptables.

A. Menambah Iptables

Perancangan proses menambah *rules* iptables dapat dijelaskan dengan *flowchart* berikut:



Gambar 3.7 *flowchart* add iptables

Penambahan *rules* iptables dimulai dengan pendeteksian VLAN name dan net id yang telah dibuat. Pendeteksian VLAN didapat dari file yang dibuat pada halaman add VLAN. Setelah diambil variable tersebut membuat combo box. Combo box berisi VLAN name dengan value adalah net id.

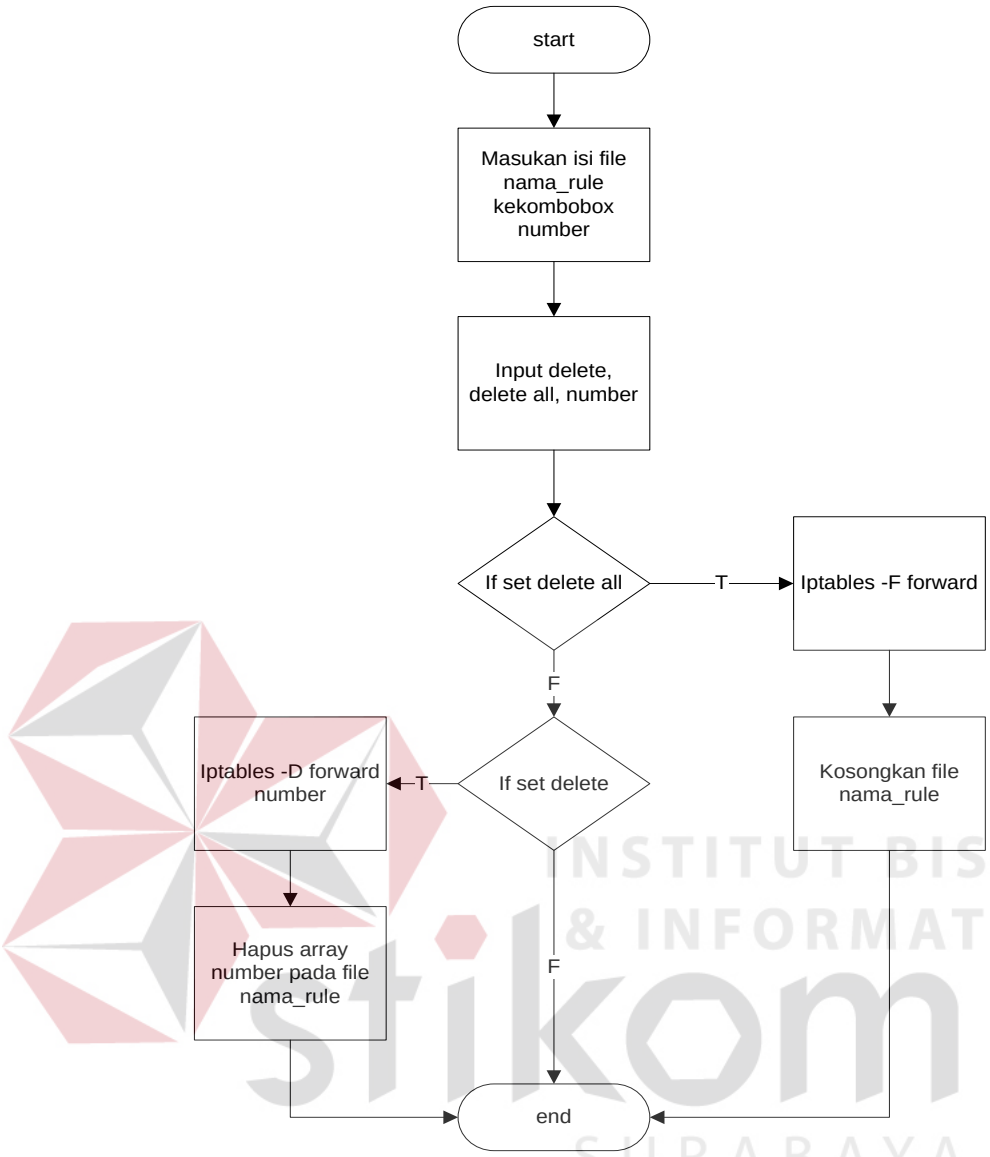
Input nat disini bersifat opsional, bila nat diisi maka program menjalankan perintah iptables –t nat –A *POSTROUTING* –o nat. hal ini berarti semua VLAN yang dibuat, menggunakan jaringan nat sebagai koneksi internet. Input nat diisi dengan interface jaringan yang terkoneksi dengan internet.

User akan memasukan inputan ip asal, ip tujuan ,nat, combo box VLAN name serta *chain*. Halaman ini mendeteksi salah satu, bila *source* terisi maka input *source* yang terpakai, selanjutnya menjalankan perintah iptables –A *FORWARD* –s *source* –d *destination* –j *chain*.

sebaliknya bila combo box VLAN name yang terisi, makan input dari kombo box yang terpakai. Selanjutnya program menjalankan perintah iptables –A *FORWARD* –s net id –d *destination* –j *chain*.

B. Menghapus Iptables

Perancangan proses menambah *rules* iptables dapat dijelaskan dengan *flowchart* berikut:

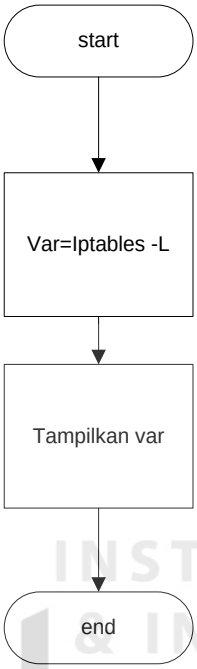


Gambar 3.8 *flowchart* penghapusan iptables

untuk menghapus *rule*, *user* harus memilih combo box *delete* ataupun *delete* all, bila *delete* yang dipilih maka *user* harus memasukan input baris yang dihapus. Pertama *user* harus memasukan input, input *delete*all yang dipilih maka PHP memanggil command iptables –F *FORWARD*, jika *delete* yang dipilih maka PHP memanggil command iptables –D *FORWARD* baris.

C. Melihat Iptables

Perancangan proses melihat *rules* iptables dapat dijelaskan dengan *flowchart* berikut:



Gambar 3.9 *flowchart* melihat iptables

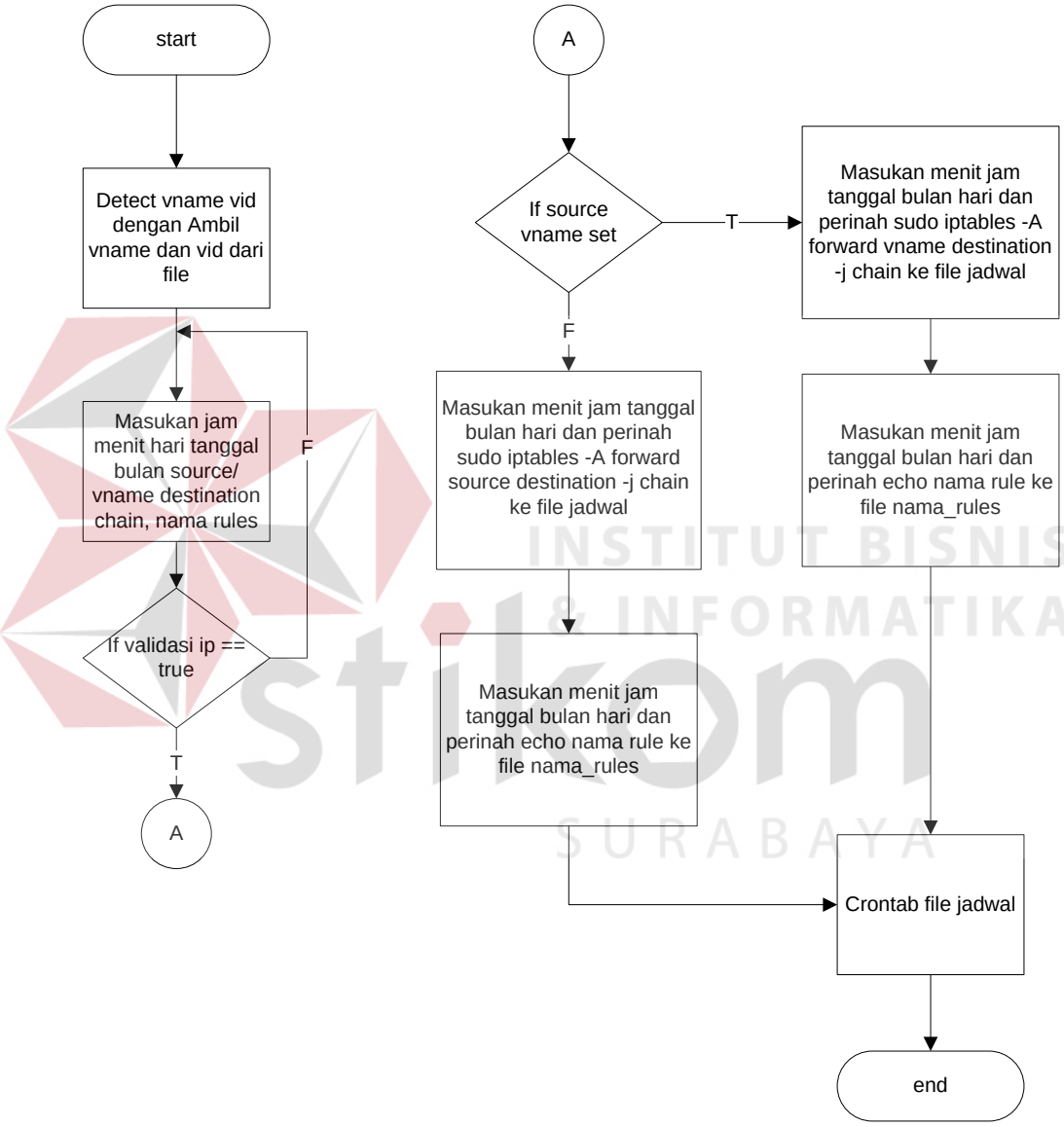
Pada *flowchart* diatas, cara yang digunakan adalah memasukan isi dari perintah iptables –L dan kemudian memasukannya dalam suatu variabel. Variabel tersebut kemudian ditampilkan ke halaman tersebut.

3.2.3 Pengaturan Penjadwalan Iptables

Pengaturan penjadwalan iptables terbagi ada 4 halaman PHP, yaitu halaman untuk membuat jadwal penambahan *rules*, halaman untuk menghapus jadwal penghapusan *rules*, halaman untuk melihat jadwal dan halaman untuk menghapus jadwal.

A. Pembuatan Jadwal Penambahan *Rules*

Perancangan proses membuat jadwal penambahan *rules* dapat dijelaskan dengan *flowchart* berikut:



Gambar 3.10 *flowchart* penjadwalan penambahan *rules*

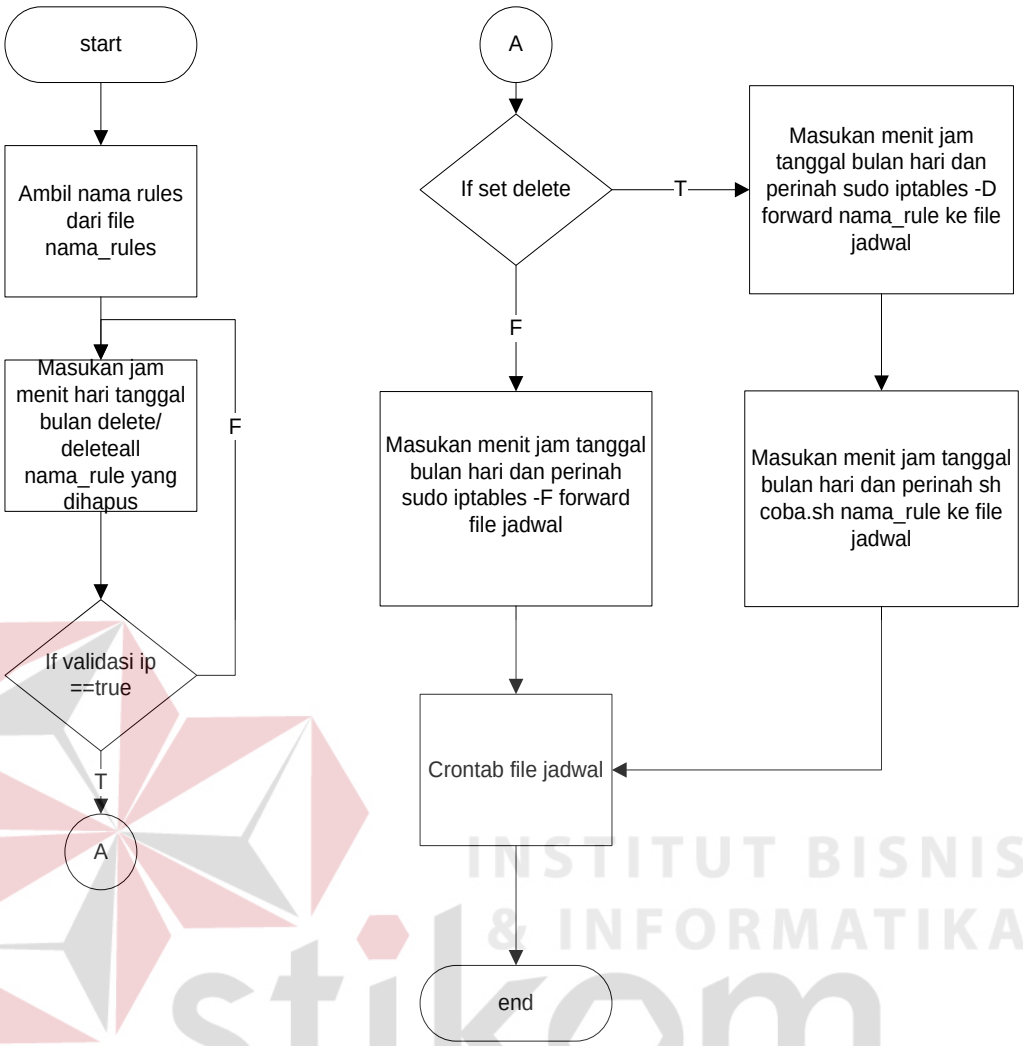
Penjadwalan penambahan *rules* iptables mempunyai konsep yang hampir sama dengan penambahan iptables. Penjadwalan dimulai dengan pendeteksian VLAN name dan net id yang telah dibuat. Pendeteksian VLAN didapat dari file

yang dibuat pada halaman add VLAN. Setelah diambil variable tersebut membuat combo box. Combo box berisi VLAN name dengan value adalah net id.

User harus memasukan input penjadwalan, *source* (dari kombo box ataupun masukan manual), *destination* dan juga *chain*. Setelah mengisi inputan, yang dilakukan program adalah memroses hasil inputan tersebut. Bila combo box terisi, maka program mengabaikan inputan *source* dan program menyimpan teks ke file jadwal dengan format [menit] [jam] [tanggal] [bulan] [hari] sudo iptables –A *FORWARD* –s netid yang dipilih –d *destination* –j *chain*. Sebaliknya bila *source* terisi maka program menyimpan teks ke file jadwal dengan format [menit] [jam] [tanggal] [bulan] [hari] sudo iptables –A *FORWARD* –s *source* –d *destination* –j *chain*. Setelah itu menjalankan perintah crontab file jadwal.

B. Pembuatan Jadwal Penghapusan *Rules*

Perancangan proses membuat jadwal penghapusan *rules* dapat dijelaskan dengan *flowchart* berikut:

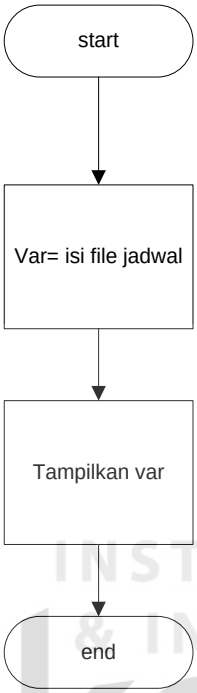


Gambar 3.11 *flowchart* menghapus jadwal *delete rules*

Inputan *user* pada halaman ini hampir sama dengan halaman sebelumnya, *user* memasukan menit, jam, tanggal, bulan, hari, *delete*/*delete*all, dan baris. Inputan baris berfungsi bila perintah *delete* yang dipilih. Bila *delete* yang dipilih maka program memanggil perintah * * * * * sudo iptables –D *forward* baris, * * * * * merupakan inputan jadwal yang dimasukan *user*. Sebaliknya bila *delete* all yang dipilih maka program memanggil perintah * * * * * sudo iptables –F *FORWARD*.

C. *List* Penjadwalan

Perancangan proses melihat penjadwalan dapat dijelaskan dengan *flowchart* berikut:

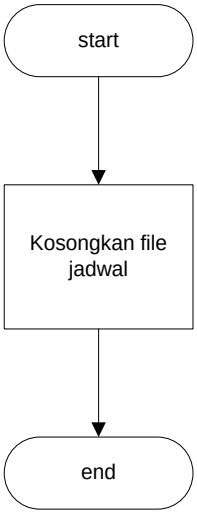


Gambar 3.12 *flowchart* menampilkan penjadwalan

Untuk melihat penjadwalan dapat dilakukan dengan menangkap isi file jadwal pada sebuah variabel, dengan perintah var=shell_exec(cat file jadwal). Kemudian langkah berikutnya adalah menampilan var.

D. Hapus Penjadwalan

Perancangan proses menghapus jadwal dapat dijelaskan dengan *flowchart* berikut:



Gambar 3.13 *flowchart* menghapus jadwal

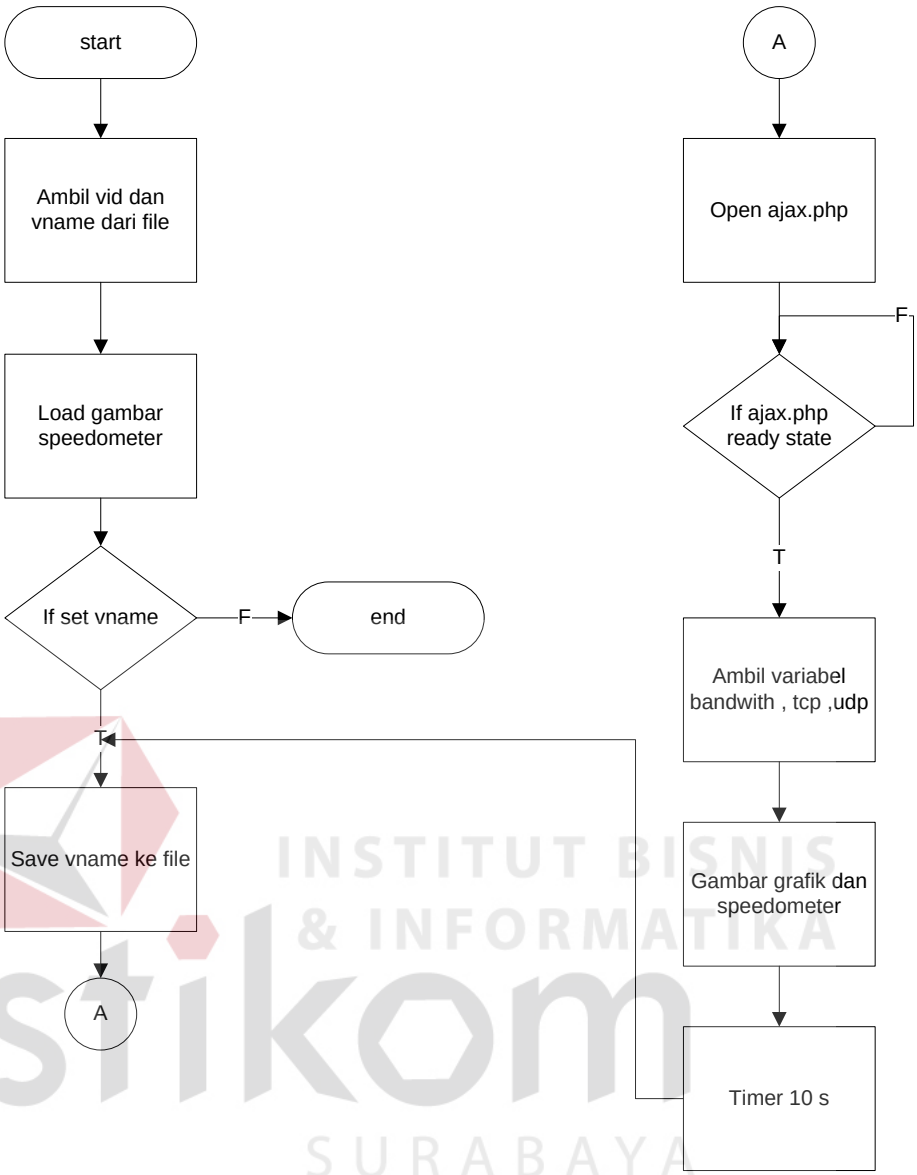
Untuk menghapus jadwal, berarti menghapus file jadwal. Hal ini dilakukan dengan cara mengisikan karakter kosong kedalam file tersebut. Perintah yang digunakan adalah shell_exec(echo “” > file jadwal).

3.3 Perancangan Monitoring pada PC *Router*

Perancangan monitoring meliputi 2 hal yaitu monitoring *bandwith* tiap VLAN dan monitoring koneksi PC laboratorium dengan jaringan PC *Router*.

3.3.1 Monitoring *Bandwith*

Untuk algoritma program monitoring *bandwith* yang digunakan akan dijelaskan melalui *flowchart* berikut:



Gambar 3.14 *Flowchart* menampilkan *bandwith* dan grafik bar

Monitoring dimulai dengan pendeteksian VLAN name dan net id yang telah dibuat. Pendeteksian VLAN didapat dari file yang dibuat pada halaman add VLAN. Setelah diambil, variable tersebut membuat combo box. Combo box berisi VLAN name dengan value adalah net id. Berikutnya program menampilkan gambar speedometer, dan merequest file ajax untuk mendapat parameter,

parameter nantinya digunakan untuk menggambar grafik bar tiap *port* dan juga menggapdate variabel dari speedometer.

User memilih nama VLAN yang akan dimonitoring, jika program mendeteksi pemilihan maka variabel net id disimpan di file pemilihan. Setelah itu memanggil request untuk membuka file ajax.php. berikut perintah untuk merequest http, `var xmlhttp=new XMLHttpRequest();`

`xmlhttp.open("GET","ajax.php",true);`

setelah membuka file ajax, program akan menunggu file ajax sampai terbuka dengan sempurna (ready state = 4). Nilai ready state mempunyai 5 nilai. Berikut ini adalah daftar lengkap dari nilai ready state: 0 (uninitialized), 1 (Loading) 2 (Loaded) 3(interactive) 4(complete). File dibaca dengan baik bila state file adalah 4 dan status bernilai 200, setelah itu hasil dari file ajax diambil. Parameter yang diambil adalah persentasi masing-masing *port*, baik TCP, UDP, ICMP maupun *bandwith*. Kemudian parameter dipresentasikan dengan gambar dan grafik. Berikut adalah cuplikan kode agar menunggu sampai file terbuka dengan sempurna.

```
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status == 200)
    {
      parameters = xmlhttp.responseText.split('|');

      for(var x=0;x<47;x++)
      {
            for(var y=0;y<2;y++)
            {
                  nama[x][y]=parameters[(x*2)+2+y];
                  alert(nama[x][y]);
            }
      }
}
```

Program diatas berfungsi untuk memanggil fungsi setiap ready state berubah, dan data akan diambil bila ready state bernilai 4 yaitu complete dan status code bernilai 200. Status code 200 berarti data telah termuat dengan sempurna. Pengambilan data dimasukan dalam variabel parameter. Data yang masuk adalah data yang ditampilkan pada file ajax dengan pemisah (|). Data yang masuk pada parameter dimasukan kedalam array 2 dimensi agar dapat dimanupulasi dengan mudah, dengan nama(x)(0) berarti nama *port* dan (x)(1) berarti besar presentase.

Menggambar grafik dilakukan dengan membuat tabel. Untuk grafik vertikal, program membuat 1 baris dengan banyak kolom. Kolom diisi dengan jumlah presentase *port* + (peresentase *port* + 1 pixel) dengan style background berwarna merah + nama *port*. Penggambaran ini dilakukan berulang-ulang sampai setiap variabel tergambar. Berikut adalah cuplikan file menggambar grafik secara vertikal.

```
var html = '';
      html += '<table border="0"><tr>';
      for (var i=0; i < 14; i++)
      {
html   +=   '<td   align="center"   valign="bottom">'      +
parseInt(nama[i][1])                    +                    '<div
style="background:red;width:30px;height:'                       +
(parseInt(nama[i][1])+1)+'px"> </div>' + nama[i][0] +'</td>';
      }
        html += '</tr></table>';
      document.getElementById("graphDiv").innerHTML = html;
```

sintak diatas merupakan sintak java*script* sehingga setiap penulisan sintak html harus ditambahkan dengan ". perulangan dari 0 sampai 14 merupakan pembuatan data vertikal dari paket TCP. Tiap perulangan proses yang ditampilkan adalah presentasi digabungkan dengan 1+presentase dalam pixel digabungkan dengan nama *port*. Setelah itu dituliskan pada graphDiv.

Untuk menggambar horisontal **,** program membuat 1 baris dengan 3 kolom. Kolom pertama diisi dengan jumlah presentase *port*, kolom kedua diisi (peresentase *port* + 1 pixel) dengan style background berwarna merah dan kolom ketiga diisi dengan nama *port*. Penggambaran ini dilakukan berulang-ulang sampai setiap variabel tergambar. Berikut adalah cuplikan file menggambar grafik secara horisontal.

```
var html2 = '';
      html2 += '<table border="0">';
      for (var i=15; i < 20; i++)
      {
        html2+= '<tr>';
        html2+= '<td>';
        html2+= nama[i][0];
        html2+= '</td>';
        html2+= '<td>';
        html2+= '<hr id="red" style="height: .8em; '
        + 'background-color: green; color: green; '
        + 'margin-left: 0; text-align: left; '
          + 'width: ' + (parseInt(nama[i][1])+1) + 'px">';
        html2+= '</td>';
        html2+= '<td>';
        html2+= nama[i][1];
        html2+= '</td>';

      }
      html2 += '</table>';
document.getElementById("graph").innerHTML = html2;
```

Prinsipnya hampir sama dengan menggambar secara vertical. Penulisan html dimasukan pada variabel html 2. Perulangan dilakukan sebanyak 5 kali, sesuai dengan jumlah paket *port* paket UDP dari data ke 15 sampai data ke 20. Untuk menggambar, dibuat tabel dengan setiap baris mempunyai 3 kolom, kolom pertama berisi keterangan *port*, kedua berisi panjang *port* dalam pixel yang berwarna hijau dan ketiga berisi panjang *port* dalam angka.

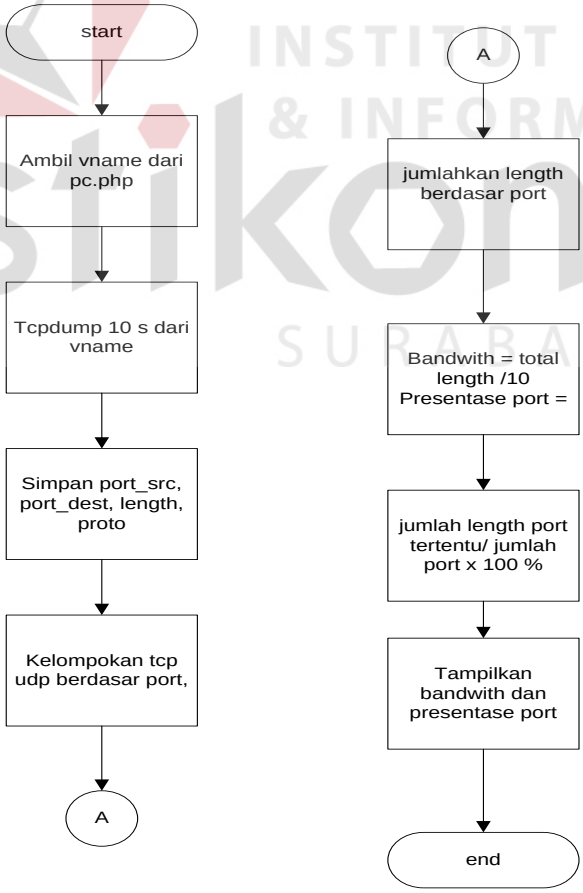
Untuk Menggambar speedometer dilakukan dengan meload gambar speedometer dari bindow *script*. Jika ada perubahan parameter *bandwith*, maka variabel bantu akan bertambah atau berkurang 1 sampai sama dengan variabel

bandwith. Setiap penambahan dan pengurangan variabel bantu, digambarkan secara langsung oleh pergerakan jarum pada speedometer. Berikut cuplikan penggambaran speedometer pada program.

```
	var	gauge	=	bindows.loadGaugeIntoDiv("gauge.xml",
"gaugeDiv");
	if (simpan > parseInt(parameters[1]))
		{
			drawGaugemin();

		}
		else
		{
			drawGauge();

		}
	function drawGaugemin()
		{
	;
		if (simpan-1<parseInt(parameters[1]))
		{
			simpan=parseInt(parameters[1]);
		}
		else
		{
			simpan=simpan-1;
			updateGauge();
			var t2=setTimeout("drawGaugemin()",100);
		}
		}

		function drawGauge()
		{


		if (simpan+1>parseInt(parameters[1]))
		{
			simpan=parseInt(parameters[1]);
		}
		else
		{
			simpan=simpan+1;
			updateGauge();
			var t2=setTimeout("drawGauge()",100);
		}
		}


		function updateGauge() {

			gauge.needle.setValue(simpan);
			gauge.label.setText( simpan );
		}
```

Var gauge merupakan variabel untuk memanggil speedometer yang dibuat oleh bindow. Simpan adalah variabel bantu yang digunakan untuk menggambarkan jarum pada speedometer. Bila simpan lebih kecil dari *bandwith* maka, akan memanggil fungsi pengurangan simpan (draw gauge min). Sebaliknya jika simpan lebih besar makan simpan akan bertambah satu persatu sampai tercapai hasil yang sama dengan *bandwith* (draw gauge). Setiap pengurangan atau penambahan pada variabel simpan akan digambarkan pada fungsi *update* gauge. Dimana *update* gauge menggambarkan posisi jarum dan text pada speedometer.

Berikut adalah *flowchart* dari file ajax, file ajax adalah file yang mengolah data dari TCP dump menjadi data matang berupa *bandwith* dan presentase tiap *port*:



Gambar 3.15 *flowchart* file ajax untuk mengolah data monitoring

Pertama file ajax membaca variabel yang telah dibuat oleh monitoring dari file pemilihan, kemudian program memberikan perintah TCPdump –n –v dengan interface variabel selama 10 detik. Opsi –n berguna untuk menghilangkan sistem dns, sehingga yang terlihat adalah ip bukan nama dari name server, sedangkan –v berguna untuk melihat paket secara lengkap termasuk panjang paket. Berikut adalah cuplikan dari TCP dump.

```
[cc lang="VHDL"]19:57:06.748557 IP (tos 0×0, ttl 64, id 33646,
offset 0, flags [DF], proto TCP (6), length 60)
192.168.1.4.33922 > 68.178.254.190.80: Flags [S], cksum 0x83ae
(correct), seq 4011514848, win 5840, options [mss 1460,sackOK,TS
val 612494 ecr 0,nop,wscale 6], length 0[/cc]
```

Keterangan:

1. 19:57:06.748557 merupakan waktu diambilnya TCPdump
2. tos 0×0 => tipe servis
3. ttl 64 merupakan *time* to live
4. id 33646 menunjukan nomer id paket
5. [DF] berarti don't *fragment*, artinya paket yang dikirim tidak terbagi-bagi.
6. *proto* TCP merupakan tipe dari *protocol*, bisa UDP, TCP ataupun ICMP
7. *length* 60 panjang paket + *header*
8. 192.168.1.4.33922 192.168.1.4 merupakan ip sumber dan 33922 *port number* yang digunakan oleh *client*
9. 68.178.254.190.80 , ip tujuan dengan *port* 80
10. *Flags* [S] merupakan *flag* dari TCP dimana parameter S berarti ack *reply* dari server, parameter r berarti koneksi diulang(reset), parameter F berarti untuk penanda akir dari pengiriman data, dan parameter P berarti data *harus* dikirim secepatnya.
11. cksum 0x83ae (correct) merupakan TCP-*header* check-sum dari paket.

12. seq 4011514848 merupakan TCP sequence *number*
13. win 5840 merupakan jumlah yang akan dikirim sebelum mendapat Paket ACK dikirim kembali dari server
14. *length* 0 merupakan panjang dari data, bernilai 0 karena yang dikirim adalah *header*.

Setelah dijalankan Data dari tcpdump disimpan kedalam file hasil. program akan memotong informasi *destination port*, *source port*, panjang data + *header*, dan jenis *protocol*. Cara pemotongan yang digunakan adalah pemotongan berdasarkan kolom. Dari beberapa percobaan, diketahui bahwa kolom *protocol* berada pada kolom 14, *port source* berada pada kolom 3, panjang paket berada pada kolom 17, dan *port destination* berada pada kolom 1. Tiap data yang dipotong dimasukan kedalam file, untuk media penyimpanan. Kemudian proses selanjutnya, file dibuka, data file diambil dan dimasukkan kedalam array. Berikut cuplikan pemotongan jenis protokol pada file ajax.

```
$save1=shell_exec ("cat /home/sendy/monitor/hasil3 |grep proto |
awk '{print $14}' > /home/sendy/monitor/proto")
fopen('/home/sendy/monitor/proto', 'r');
      while (!feof($fp)) {
            $kata2=fgetc($fp);
            if ($kata2=="T"){
            $proto[$count]="TCP";
            $count=$count+1;
            }
            else if($kata2=="U"){
            $proto[$count]="UDP";
            $count=$count+1;
      }
            else if($kata2=="I"){
            $proto[$count]="ICMP";
            $count=$count+1;
            }
      }
Fclose
```

Pada sintak diatas variabel save berfungsi untuk memanggil perintah terminal untuk memotong baris ke 14 pada file hasil dan menyimpanya kedalam file proto, dimana file hasil adalah file berisi paket tcpdump selama 10 detik. Cat merupakan perintah pada terminal yang berfungsi untuk menampilkan isi file. Grep berfungsi untuk menangkap kata, sedangkan awk berarti pemotongan pada kolom.

Tiap variabel dimasukan kedalam array tertentu, sehingga terbentuk 4 variabel array dari *destination port*, *source port*, *length* dan *protocol*. Kelompokan tiap *port* berdasarkan jenis protokolnya dan jumlahkan panjang data. Suatu paket dikatakan *port* http bila *source port* atau *destination port* bernilai 80. Berikut adalah cuplikan pengolahan data file ajax:

```
for ($i=0;$i<$count;$i++){
if ($proto[$i]=="TCP"){
     if($kesatuanport[$i]==20 or $kesatuanport_s[$i]==20){
     $ftpdata=$ftpdata+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==21 or $kesatuanport_s[$i]==21){
     $ftp=$ftp+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==22 or $kesatuanport_s[$i]==22){
     $ssh=$ssh+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==23 or $kesatuanport_s[$i]==23){
     $telnet=$telnet+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==25 or $kesatuanport_s[$i]==25){
     $smtp=$smtp+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==53 or $kesatuanport_s[$i]==53){
     $nameserver=$nameserver+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==80 or $kesatuanport_s[$i]==80){
     $http=$http+$kesatuanpanjang[$i];
     }
```

```
else if($kesatuanport[$i]==109 or $kesatuanport_s[$i]==109){
     $pop2=$pop2+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==110 or $kesatuanport_s[$i]==110){
     $pop3=$pop3+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==220 or $kesatuanport_s[$i]==220){
     $imap3=$imap3+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==443 or $kesatuanport_s[$i]==443){
     $https=$https+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==993 or $kesatuanport_s[$i]==993){
     $imaps=$imaps+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==995 or $kesatuanport_s[$i]==995){
     $pop3s=$pop3s+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]>1023 and $kesatuanport_s[$i]>1023){
     $others=$others+$kesatuanpanjang[$i];
     }
}
else if ($proto[$i]=="UDP"){
     if($kesatuanport[$i]==53 or $kesatuanport_s[$i]==53){
     $u_nameserver=$u_nameserver+$kesatuanpanjang[$i];
     }
else if($kesatuanport[$i]==80 or $kesatuanport_s[$i]==80){
     $u_http=$u_http+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==138 or $kesatuanport_s[$i]==138){
     $netbios_dgm=$netbios_dgm+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==161 or $kesatuanport_s[$i]==161){
     $snmp=$snmp+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==220 or $kesatuanport_s[$i]==220){
     $u_imap3=$u_imap3+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]==443 or $kesatuanport_s[$i]==443){
     $u_https=$u_https+$kesatuanpanjang[$i];
     }

else if($kesatuanport[$i]>1023 and $kesatuanport_s[$i]>1023){
     $u_others=$u_others+$kesatuanpanjang[$i];
     }

else if ($proto[$i]=="ICMP"){
     $ICMP=$ICMP+$kesatuanpanjang[$i];
     }
```

```
        $total=$total+$kesatuanpanjang[$i];
}
```

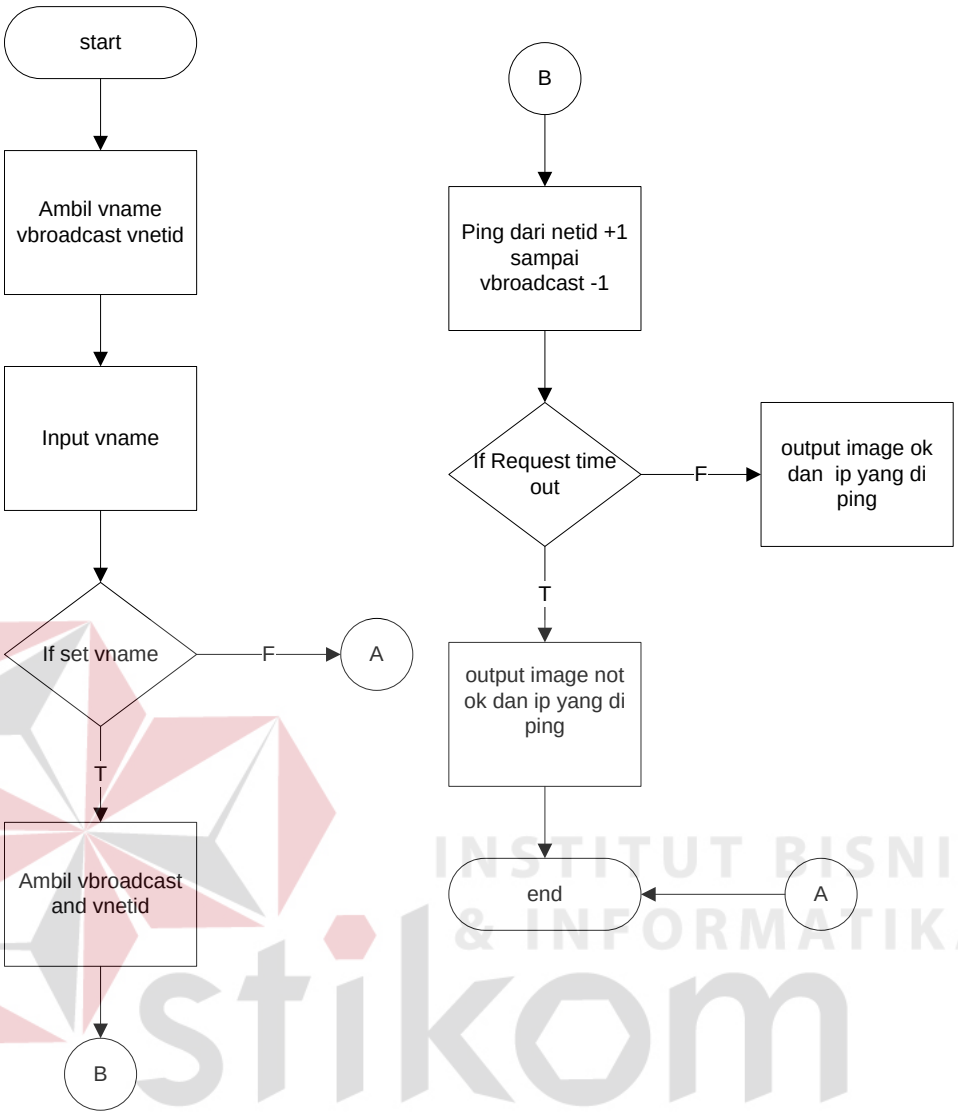
Pada sintak diatas terlihat bahwa setiap paket yang masuk didiurutkan berdasarkan jenis *protocol*, baik *protocol* TCP, UDP dan ICMP. Jika *port* TCP, dikelompokan lagi berdasarkan nomor *port*, suatu paket dikatakan *port* http jika *source port*nya bernilai 80 atau *destination port* bernilai 80, begitu juga dengan *port* lainya. Jika masuk ke pengelompokan *port* http, maka jumlah dari data dimasukan kedalam variabel tertentu.

Jenis *port* yang ditampilkan adalah *port* yang umum yaitu untuk *protocol* TCP adalah *port* ftp, ftp data, ssh, telnet, smtp, nameserver, http, pop2, pop3, imap3, https, imaps, pop3s, dan other. Perlu diketahui others merupakan *port* diluar *well-known-port*. Sedangkan untuk UDP dengan *port* nameserver, http, https, netbios dgm, snmp, imap3, others. Dan yang terakhir adalah *protocol* ICMP. *Protocol* ICMP tidak mempunyai *port number*s.

Bandwith didapat dengan hasil penjumlahan total panjang tiap *port* dibagi dengan 10 (TCPdump berjalan selama 10 detik). Panjang data pada TCP dump mempunyai satuan byte. Sehingga perlu dibagi 1000 lagi untuk menjadi KiloBytes/second. Sedangkan presentase masing-masing *port* didapat jumlah panjang tiap *port* tersebut dibagi jumlah total *port* dikali dengan 100 persen.

3.3.2 Monitoring Koneksi PC dengan PC *Router*

Monitoring koneksi PC yang dilakukan menggunakan metode ping ke tiap PC dari jaringan yang dipilih, berikut adalah *flowchart* monitoring koneksi PC:

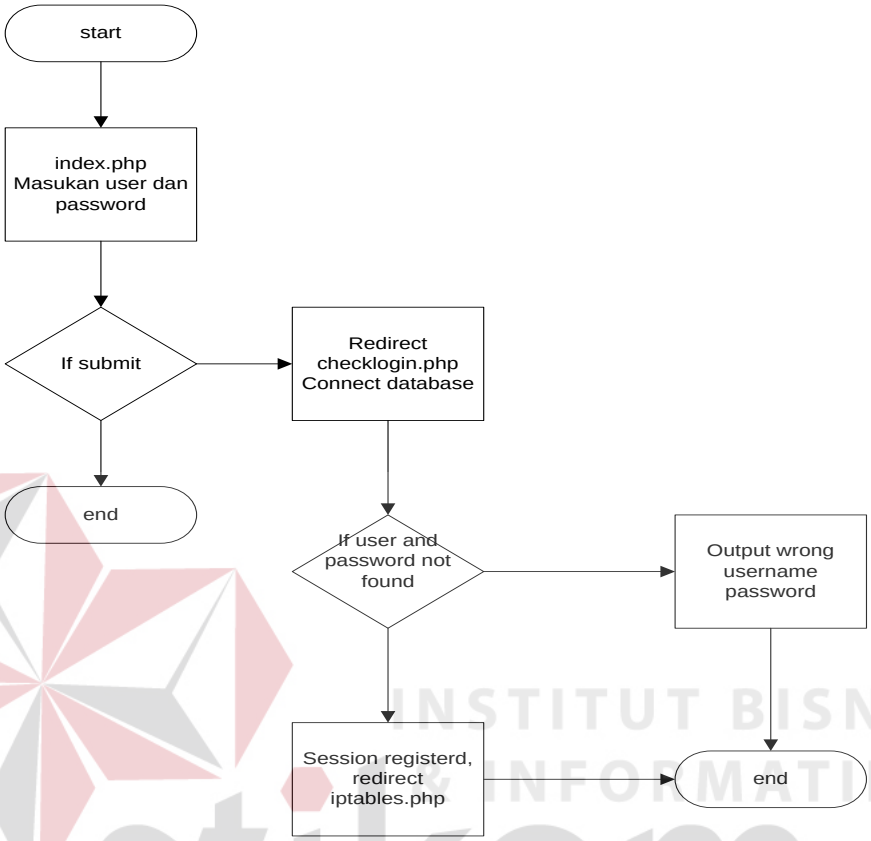


Gambar 3.16 *flowchart* monitoring PC

Pertama-tama program mengambil dari file yang dibuat oleh pembuatan VLAN, file yang diambil adalah VLAN name, VLAN *broadcast*id dan VLAN netid. Algoritma dari pemrograman ini adalah jika VLAN name terpilih dari kombo box, maka program memberikan perintah untuk mengeping dari netid +1 sampai dengan *broadcast* id -1 dari VLAN name tersebut. Jika balasan ping adalah *reqest time out* maka menampilkan gambar sukses dan menampilkan ip yang di ping, jika tidak *request time out* maka menampilkan gambar gagal dan menampilkan ip yang di ping.

3.4 perancangan authentifikasi

Perancangan authentifikasi akan dijelaskan dengan *flowchart* berikut:



Gambar 3.17 *Flowchart* authentifikasi

Pada index PHP, terdapat form login dimana *user* harus memasukan *user*name login dan password login. Authentifikasi dibuat agar orang yang berwenang saja yang dapat mengakses PC *Router*. Setelah di submit form, isi dari *user*name dan password disimpan dan page akan merubah ke halaman checklogin, pada halaman checklogin, program akan melakukan koneksi ke database, jika *user*name dan password tidak berada di database maka akan menampilkan *wrong username password*. Jika benar maka session akan terdaftar kemudian halaman page akan dirubah lagi ke iptables.php.

Berikut adalah cuplikan sintak dari checklogin.php:

```
$sql="SELECT * FROM $tbl_name WHERE username=root and
password='$mypassword'";
$result=mysql_query($sql);
$count=mysql_num_rows($result);
if($count==1){
session_register("myusername");
session_register("mypassword");
header("location:ip-tables.php");
}
else {
echo "Wrong Username or Password";
}
```

Baris pertama merupakan penyimpanan penyimpanan isi table members mysql kedalam variabel sql. Variabel count akan mengecek kesamaan *user*name dan password pada database. Bila cocok, maka nilai count bernilai 1, dan session terdaftar, jika tidak bernilai 1 maka *user* dianggap tidak terdaftar.

Untuk mencegah *user* masuk ke aplikasi tanpa authentifikasi, maka tiap awal page dari tiap fungsi diberi sintak. Isi sintak tersebut adalah, page tersebut dapat dibuka bila session teregistrasi (sudah login terlebih dahulu). Berikut adalah cuplikan sintak yang harus dimasukan pada tiap halaman:

```
session_start();
if(!session_is_registered(myusername)){
header("location:index.php");
}
```

Sintak diatas berguna untuk menolak *user* yang langsung masuk ke halaman fungsi tanpa login terlebih dahulul. bila session tidak terdaftar melalui proses login, maka diredirect ke file index.php

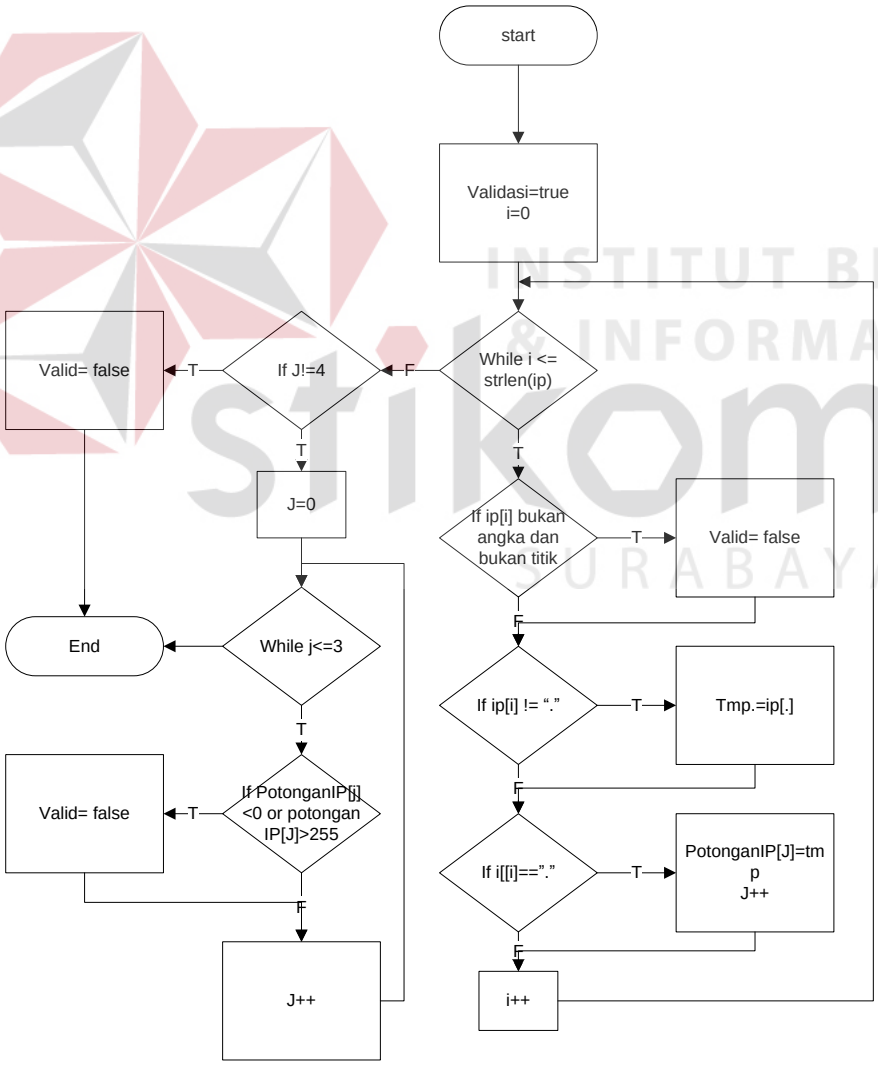
Untuk page log out, page hanya keluar dari session dan kembali ke halaman login, berikut cuplikan sintak untuk log-out.php:

```
session_start();
session_destroy();
     header("location:index.php");
```

sedangkan untuk database, dilakukan dengan mysql. Database yang digunakan diberi nama *user*name dan tabelnya bernama members. Untuk masuk kedatabase menggunakan sintak. Mysql –u root –p. sehingga hanya root sajalah yang dapat masuk kedalam database mysql yang telah dibuat.

3.5 Perancangan Validasi IP

Perancangan Validasi terhadap IP dapat dijelaskan dengan flowchart berikut:



Gambar 3.18 *Flowchart* validasi IP

Untuk validasi, ip dikatakan benar bila mempunyai format xxx.xxx.xxx.xxx, dimana tiap xxx berisi angka bulat dengan kisaran antara 0-255. Ip dikatakan salah, bila tidak sesuai dengan format diatas.

Validasi IP dilakukan dengan mendeteksi inputan ip dari *user*. Program akan mendeteksi karakter yang dimasukan *user* tiap karakter. Bila karakter yang dimasukan *user* bukan angka dan bukan titik, maka validasi akan dibuat menjadi salah, bila angka, kemudian program akan menghitung panjang angka tersebut, bila angka tersebut mempunyai panjang lebih besar 3 digit, maka validasi bernilai salah, sebaliknya jika mempunyai panjang 3 atau kurang dari 3 digit, angka dimasukan kedalam array sementara. Kemudian program akan mengecek tiap potongan 3 digit tersebut, bila 3 digit lebih dari 255, atau kurang dari 255. Maka validasi juga bernilai salah.