SISTEM *MONITORING* JARINGAN PADA SERVER LINUX DENGAN MENGGUNAKAN *SMS GATEWAY*

Taufan Dwi Prayogo[1], Kushartantya[2], Helmie Arif Wibawa[2]
Program Studi Teknik Informatika FMIPA UNDIP

ABSTRAK

Administrator jaringan bertanggung jawab melakukan *monitoring* jaringan terhadap suatu jaringan komputer. *Monitoring* harusnya dilakukan secara terus menerus untuk menjaga kualitas jaringan, karena kesalahan pada jaringan tidak bisa dideteksi kapan terjadinya. Hal ini akan menyulitkan jika administrator berada jauh dari pusat pengawasan. Situasi ini bisa berakibat fatal jika terjadi kesalahan jaringan yang tidak diketahui karena administrator jaringan sedang tidak ada di tempat pengawasan. Dengan memanfaatkan teknologi *SMS gateway* yang menggunakan server berbasis Linux, administrator jaringan dapat mendapatkan informasi kondisi server secara cepat. Hasil yang didapatkan adalah sebuah sistem yang mampu membantu administrator jaringan dalam melakukan monitoring jaringan kapan saja dan dimana saja untuk menjaga jaringan akan tetap stabil walaupun administrator jaringan sedang tidak di pusat pengawasan.

Kata Kunci: *Monitoring* jaringan, *SMS gateway*, Server Linux.

1. Pendahuluan

Pesatnya kemajuan jaman menuntut semua orang untuk selalu *update* segala informasi dari berbagai sumber. Internet merupakan salah satu sumber informasi yang paling banyak digunakan masyarakat saat ini. Kemajuan itu pula yang membuat masyarakat sudah dapat mengakses ke internet dengan berbagai cara dan tempat. Didukung dengan banyaknya Internet *Service Provider* (ISP) yang menyediakan layanan internet secara cepat membuat makin mudah dan murah dalam menggunakan internet.

Sebuah jaringan lokal yang terhubung dengan internet membutuhkan server yang dapat berfungsi sebagai *router* maupun *gateway*, sehingga komputer *client* dalam jaringan tersebut dapat melakukan komunikasi secara *online*. Sebuah server dituntut untuk dapat memberikan layanan real time secara 24 jam. Terkadang karena banyaknya user yang mengakses melalui sebuah server menyebabkan kepadatan arus jalur data pada jam-jam tertentu yang menyebabkan gagal koneksi atau *Request Time Out.*

Dalam pengelolaan sebuah server dibutuhkan seseorang yang memiliki kemampuan serta tanggung jawab yang tinggi dalam menjamin server tersebut dapat melayani komputer *client* dengan baik yang biasa disebut administrator jaringan. Tetapi kadang kala administrator jaringan tidak dapat mengawasi koneksi server secara 24 jam penuh, oleh karena itu untuk memperingan pekerjaan administrator jaringan dibutuhkan sebuah sistem yang dapat menginformasikan status koneksi dari server secara real time.

Dibutuhkan teknologi alternatif untuk membantu administrator jaringan mendapatkan informasi mengenai status koneksi jaringannya secara cepat dan akurat. Teknologi yang mungkin dapat digunakan adalah teknologi SMS, SMS tidak hanya dapat digunakan untuk berkomunikasi secara cepat dan murah, namun dapat dimanfaatkan dalam banyak hal. Salah satunya sebagai jaringan alternatif penghantar pesan untuk menginformasikan status koneksi server kepada admininstrator jaringan jika jaringan komputer terputus atau lumpuh. Kelebihan lain dari penggunaan SMS adalah dapat dibuat aplikasi yang menghubungkan server dengan administrator jaringan melalui jaringan *SMS gateway*, dengan tarif normal [6].

Untuk membuat teknologi SMS yang dapat membantu administrator jaringan, perlu dibuat sistem *SMS gateway*. Fungsi *SMS gateway* sebagai teknologi pendukung SMS agar dapat digunakan sebagai *alert system*. Penggunaan *SMS gateway* sebagai sistem yang melakukan pengiriman SMS secara otomatis. Sehingga administator jaringan akan menerima SMS tentang keadaan jaringannya yang dikirim melalui *SMS gateway*.

Teknologi monitoring jaringan yang sebelumnya hanya mengandalkan administrator

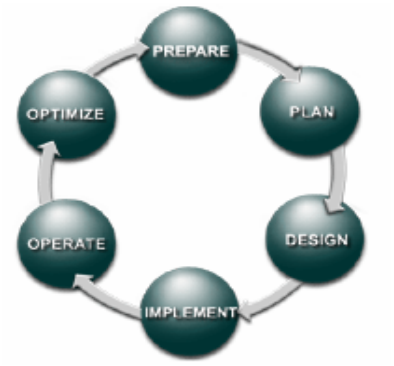
jaringan untuk memantau server jaringannya. Kekurangan dari teknologi monitoring terdahulu adalah tidak adanya sistem yang memberitahukan administrator jaringan tentang kondisi koneksi server dan sebagian besar server yang digunakan adalah server berbasis Linux. Sehingga admininstrator jaringan harus berada dalam lingkungan server agar terus dapat memantau server. Pada tulisan ini akan dibahas tentang penerapan *SMS gateway* untuk sistem *monitoring* jaringan pada server Linux yang dapat membantu tugas administrator jaringan dalam pengawasan koneksi server.

2. *Short Messaging Service*

SMS (*Short Messaging Service*) adalah suatu layanan untuk mengirimkan pesan pendek (160 karakter) dari satu telepon GSM ke yang lainnya [6]. Layanan ini juga dapat digunakan untuk mengirim pesan teks regular seperti halnya logo operator, nada dering, isi kontak telepon, dan konfigurasi telepon. Layanan SMS berisi beberapa *service* yang dapat dihubungkan melalui pesan SMS untuk beberapa nomor telepon tertentu, yang dapat menjawab isi yang diminta, apabila tersedia. Sedangkan *SMS gateway* adalah aplikasi yang menyediakan antarmuka antara pesan SMS dan protokol lainnya dan menghubungkan perangkat mobile (Ponsel, PDA phone dll) dengan SMS Center [6].

3. Tahapan *Life Cycle Approach*

Dalam pengembangan sistem berbasis jaringan, metode pengembangan sistem yang digunakan menurut [1], *Life Cycle Approach* (Pendekatan daur hidup)



Gambar 1. Tahapan-tahapan *life cycle*

a. *Prepare*
Pada bagian ini dilakukan analisa untuk menjabarkan dan menempatkan strategi *networking* yang akan dibangun sesuai dengan infrastruktur IT (*Information Technology*) dan aspek finansial yang dikembangkan ke depan.

b. *Plan*
Pada bagian perencanaan, akan dipelajari tentang infrastruktur IT (*hadware, software,* topologi jaringan) yang telah berjalan dan digunakan. Tahapan ini meneruskan dari tahap *prepare* sebelumnya. Dengan perencanaan yang baik akan membantu untuk mengatur pekerjaan, resiko yang mungkin muncul, permasalahan yang ditemui*, responsibility*, tenggang waktu *milestones*, dan kebutuhan sumber daya yang dibutukan.

c. *Design*
Dalam tahap ini menjelaskan tentang bagaimana proses konfigurasi, koneksi percobaan, pengembangan ke depan, dan proses migrasi dari sistem lama ke sistem baru, demo sistem dan validasi.

d. *Implement*
Sebelum diimplementasikan akan dilakukan *testing* terlebih dahulu, hal ini dilakukan untuk meminimalisasi kesalahan yang mungkin muncul. Proses ditahapan ini adalah instalasi, konfigurasi, dan integrasi sistem. Bisa jadi salah satu bagian *network* telah berhasil dimigrasi dengan baik namun tidak berarti semua bagian akan lancar.

e. *Operate*
Tahap ini merupakan lanjutan dari tahap implementasi, dilakukan pengawasan dan pemantauan pada pengoperasiannya, beberapa kasus pada tahapan ini akan terlihat beberapa masalah misalnya tidak kompatibelnya *hardware*, masalah pada *software* dan aplikasi yang selama ini jalan tidak ada hambatan namun setelah implementasi menjadi terganggu.

f. *Optimize*
Masukan pada saat tahapan implementasi dan *operate* akan sangat mempengaruhi tahapan optimalisasi ini, masukan tadi bisa memberikan *input* untuk penanganan, *redesign*, rekonfigurasi dan perubahan yang

perlu dilakukan tanpa merubah arah dari tujuan sistem tersebut.

4. Perancangan

Dalam pembangunan sistem *monitoring* jaringan dengan *SMS gateway* ini dibutuhkan tiga desain fungsional. yang pertama digunakan untuk menjelaskan urutan logis dalam konfigurasi sistem, kemudian kedua menggambarkan urutan logis uji koneksi ping dan yang ketiga adalah untuk uji koneksi database. Tabel berikut ini memaparkan desain fungsional konfigurasi

Tabel 3.1 Desain fungsional konfigurasi

Desain fungsional konfigurasi
i adalah jumlah host yang akan diuji dengan ping **ii** adalah jumlah host sql yang akan diuji dengan netcat **tunda** adalah jeda tiap pengujian **noa** adalah nomor ponsel admin **port** adalah port database mysql **host[]** adalah array yang berisi host yang akan diuji dengan ping **host_sql[]** adalah array yang berisi host sql yang akan diuji dengan netcat

Inisialisasi variabel
i, ii, tunda, noa, port, host[], host_sql[]
for (m = 1; m <= i; m++) {
iphost=host[m]
eksekusi file ptest.sh }
for (n = 1; n <= ii; n++) {
iphost_sql=host_sql[n]
eksekusi file ttest.sh }

Tabel 3.2 Desain fungsional ping_test

Desain fungsional ping_test
C Merupakan *counter* untuk perulangan **B** merupakan *counter* banyaknya ping yang berhasil **G** merupakan *counter* banyaknya ping yang gagal **H** merupakan variabel untuk menampung *capture* hasil ping (berisi karakter "64" jika ping berhasil dan " " *null* jika ping gagal) **iphost** merupakan alamat *host* tujuan ping

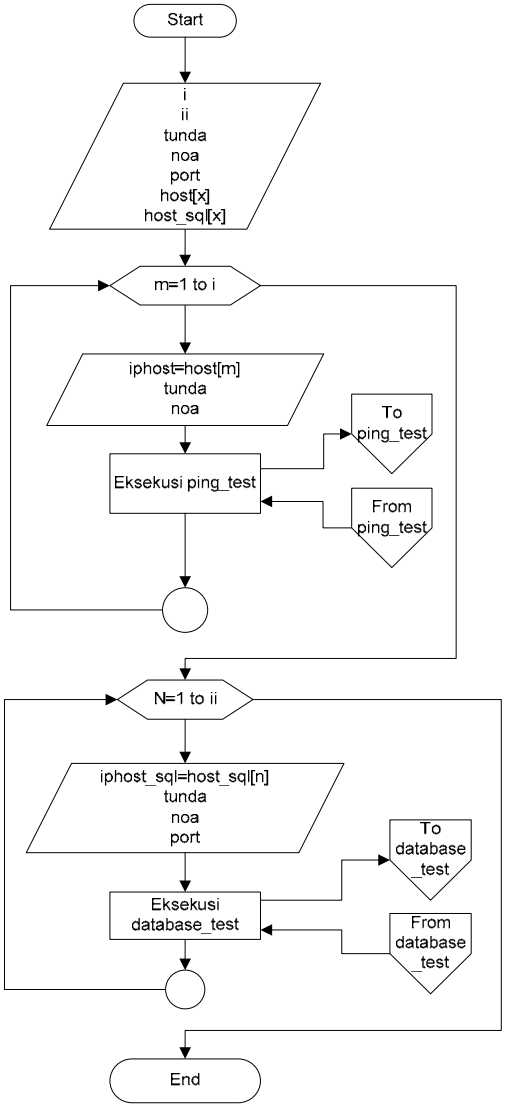
for (C = 1; C <= 10; C++) {
H=(ping $iphost)
if (H = "64") {
B=B+1
Else
G=G+1
}
}
if (G = 10) {
gammu send sms }

Tabel 3.3 Desain fungsional database_test

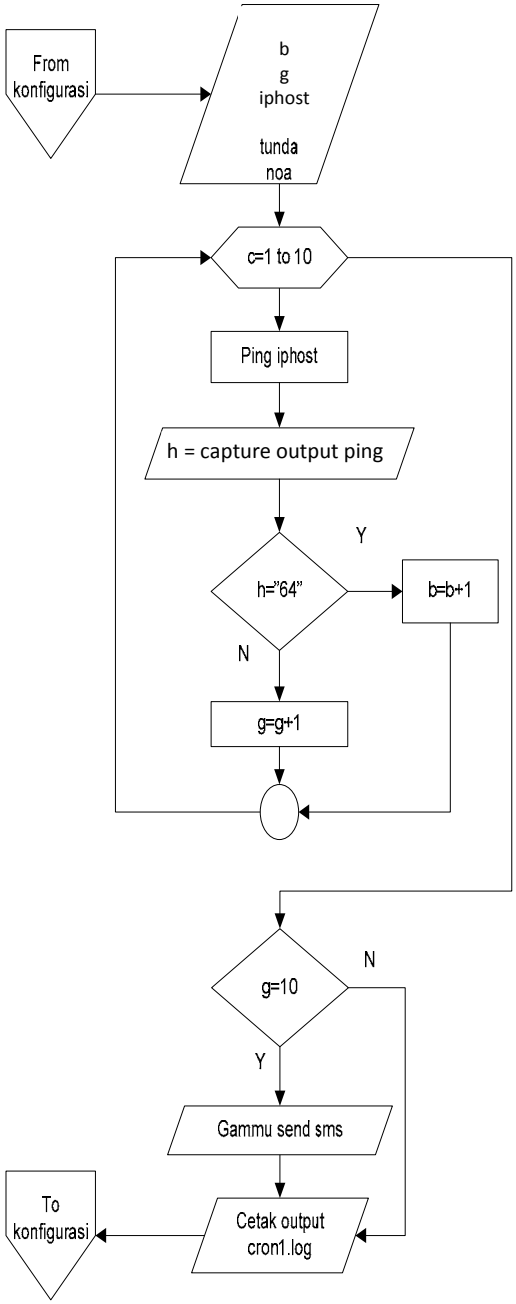
Desain fungsional database_test
C Merupakan counter untuk perulangan **B** merupakan counter banyaknya test yang berhasil G merupakan counter banyaknya test yang gagal **H** merupakan variabel untuk menampung capture hasil Netcat **iphost_sql** merupakan server tujuan database **port** merupakan port server tujuan database

for (C = 1; C <= 10; C++) {
H=(nc -zv $iphost_sql $port)
if (H = "Co") {
B=B+1
Else
G=G+1
}
}
if (G = 10) {
gammu send sms
}

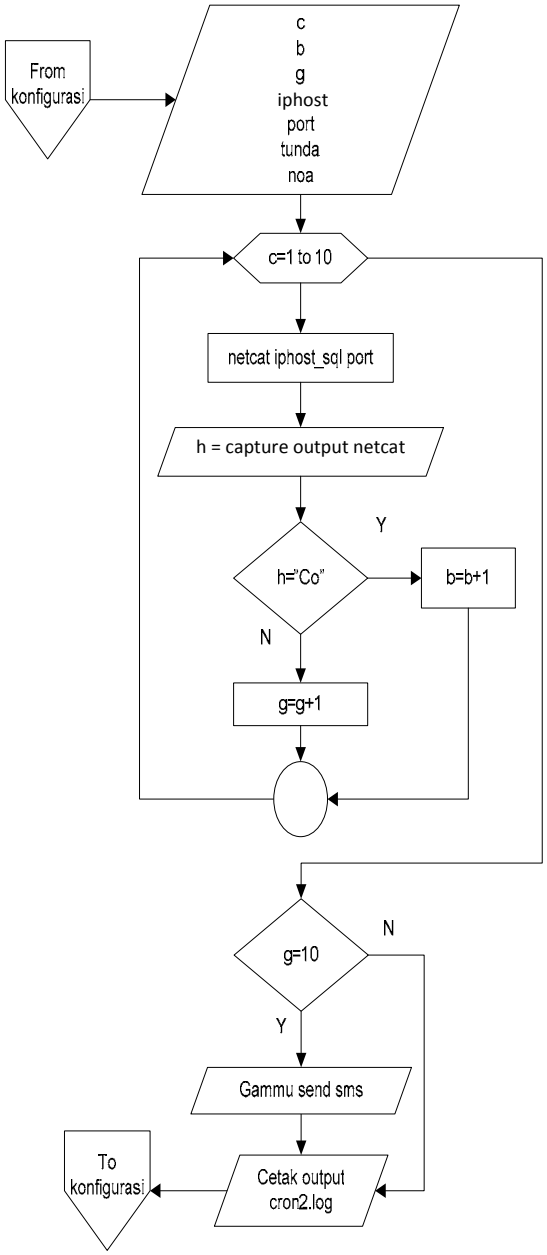
Perancangan dilakukan dengan membuat desain flowchart yang menggambarkan secara garis besar algoritma sistem, mulai dari sistem dijalankan hingga waktu terminasi.



Gambar 2. Flowchart konfigurasi



Gambar 3. Flowchart ping_test



Gambar 4. Flowchart database_test

5. Implementasi dan Pengujian Sistem

5.1. Implementasi

Langkah –langkah dalam implementasi

a. Menyiapkan direktori *script*
   Script yang dibuat nanti harus berada dalam direktori yang benar, agar sesuai manajemen direktori. Harus diletakkan pada direktori /usr/local/, direktori /usr bersifat *shareable*, yang berarti isi dari direktori /usr dapat digunakan oleh *host* lain.

b. Menyiapkan *script* uji koneksi dengan ping
   Perintah yang digunakan untuk membuat *script* uji koneksi dengan ping adalah

```
root:~# sudo gedit
/usr/local/smsmon/ptest.sh
```

   Dalam file ptest.sh, terdapat perintah untuk melakukan ping dan mengirim SMS sesuai dengan algoritma yang telah dibuat, implementasinya sebagaimana terlihat pada kode 1 berikut:

```
#!/bin/bash
# program monitoring koneksi
server dengan ping
#inisialisasi variable lokal
g=0                  #counter untuk
menghitung banyak ping yang gagal
b=0                  #counter untuk
menghitung banyak ping yang
berhasil
#Begin
for (( c = 1 ; c <= 10 ; c++ ))
do
      h=$(sleep $tunda | ping -c 1
$iphost | head-n 2|tail-n1)
      h=${h:0:2}
      jam=$(date +"%H:%M:%S")
      if [ "$h" != "64" ]
      then
            g=$((g+1))
echo "Terputus dengan $iphost pada
$jam. test ke $c. Gagal $g kali"
      else
            b=$((b+1))
            echo       "Tersambung
dengan $iphost pada $jam. test ke
$c.     Berhasil     $b     kali"
      fi
done
jam2=$(date           +"%H:%M:%S")
echo "Hasil pengujian ping server
:                                  "
echo "Berhasil    :    $b    kali"
echo "Gagal :       $g       kali"
if    [    "$g"    =    10    ]
then
      echo  "Tindakan  :  Mengirim
SMS"
      echo  "Ping  gagal,  koneksi
$iphost putus pada $jam2" | gammu-
smsd-inject       TEXT       $noa
else
```

```
      echo "Tindakan : Tidak
mengirim                    SMS"
fi
#End
```

Kode 1. Script uji koneksi dengan ping

Agar *script* ptest.sh dapat dijalankan, *script* ptest.sh harus dibuat menjadi mode *executable* [7].

c. Menyiapkan *script* uji koneksi *database* dengan Netcat
Pada tahap ini dibuat script yang berisi perintah untuk melakukan uji koneksi terhadap server *database* dan mengirim SMS sesuai dengan algoritma yang telah dibuat, implementasinya sebagaimana terlihat pada kode 2 berikut:

```
#!/bin/bash
# program monitoring port server
database Mysql dengan Netcat
#inisialisasi variable lokal
g=0   #counter untuk menghitung
banyak        gagal        netcat
b=0   #counter untuk menghitung
banyak       berhasil       netcat
#Begin
for (( c = 1 ; c <= 10 ; c++ ))
do
      h=$(sleep $tunda | nc -zv
$iphost_sql  $port  2>&1  | grep
succeeded                        >
/usr/local/smsmon/nc.log         )
      cap=$(grep         succeeded
/usr/local/smsmon/nc.log | head -
n       2|tail       -n       1)
      jam=$(date     +"%H:%M:%S")
      cap=${cap:0:2}
      if  [  "$cap"  !=  "Co"  ]
      then
            g=$((g+1))
            echo "Terputus dengan
database  $iphost_sql  pada  $jam.
test ke $c. Gagal $g kali"
      else
            b=$((b+1))
            echo       "Tersambung
dengan  database  $iphost_sql  pada
$jam.  test  ke  $c.  Berhasil  $b
kali"
      fi
done
```

```
jam2=$(date          +"%H:%M:%S")
echo   "Hasil   pengujian   server
database           :           "
echo "Berhasil    :    $b    kali"
echo "Gagal :       $g       kali"
if    [    "$g"    =    10    ]
then
      echo  "Tindakan  :  Mengirim
SMS"
      echo "Koneksi database
```

Kode 2. Script uji koneksi *database*

Script harus diubah ke dalam mode *executable* agar dapat dijalankan.

d. Membuat file konfigurasi utama sistem *monitoring*
Konfigurasi yang diperlukan sebagai informasi *host* yang akan diuji, seperti waktu tunda, nomer telpon admin, IP *Address* dan *port* menggunakan varibel global agar variabel dapat digunakan *script* lain [3]. Isi dari *script* konfigurasi ditunjukkan pada kode 3

```
#File konfigurasi sistem
monitoring jaringan
#berisi daftar host yang akan
dimonitoring
#deklarasi jumlah host
i=2
#deklarasi jumlah host sql
ii=1
#jeda tiap pengujian (detik)
export tunda=3
#nomer telpon admin
export noa=085726940720
#port database
export port=3306
#ip host
declare -a host
host[1]=192.168.1.1
host[2]=192.168.1.2
#ip host sql server
declare -a host_sql
host_sql[1]=192.168.1.2
#menjalankan uji koneksi (ping)
for (( c = 1 ; c <= i ; c++ ))
do
      export iphost=${host[$c]}
      ./ptest.sh >> cron1.log
done
```

```
#menjalankan uji koneksi database
(netcat)
for (( n = 1 ; n <= ii ; n++ ))
do
export iphost_sql=${host_sql[$n]}
      ./ttest.sh  >> cron2.log
      done
```

Kode 3. Script konfigurasi

e. Mengatur penjadwalan *script* untuk dijalankan
   Sistem *monitoring* jaringan menggunakan *SMS gateway* akan melakukan pengujian koneksi selama 10 menit sekali. Untuk mengimplementasikannya, *script* konfigurasi diatur penjadwalan eksekusinya menggunakan crontab. Caranya dengan mengetikkan perintah

```
root@:~# sudo crontab –e
```

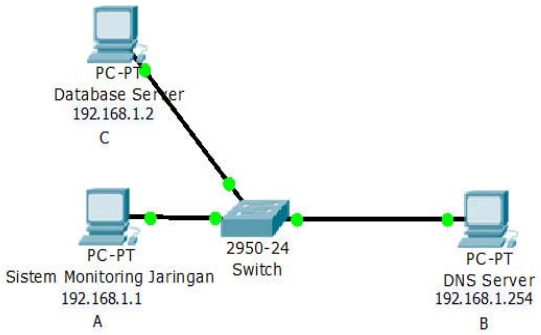
Kemudian akan terbuka editor konfigrasi crontab, selanjutnya tambahkan perintah untuk melakukan penjadwalan *script* konfigurasi

```
*/10 * * * *
/usr/local/smsmon/config.sh
```

5.2. Pengujian

Fungsi utama sistem *monitoring* jaringan berbasis *SMS gateway* adalah memberi *alert* atau peringatan kepada admin mengenai konektifitas server dengan DNS dan konektifitas *database* server secara teraktual melalui SMS [2].

Untuk melakukan pengujian, dibuat jaringan sederhana dengan topologi seperti ditunjukkan gambar 5, kemudian dilakukan percobaan dengan melakukan monitoring



Gambar 5.Topologi uji sistem *monitoring* jaringan dengan *SMS gateway*

Berdasarkan dua fitur utama dalam sistem ini, uji koneksi menggunakan ping akan dilakukan PC A kepada DNS server PC B. Jika PC A dan PC B terhubung, hasil yang diperoleh dari monitoring ini dapat dilihat pada /usr/local/smsmon/cron1.log sebagaimana terlihat pada gambar 6 berikut.

```
Tersambung pada ”16:00:06”. test
ke 1. Berhasil 1 kali
Tersambung pada ”16:00:09”. test
ke 2. Berhasil 2 kali
Tersambung pada ”16:00:12”. test
ke 3. Berhasil 3 kali
Tersambung pada ”16:00:15”. test
ke 4. Berhasil 4 kali
Tersambung pada ”16:00:18”. test
ke 5. Berhasil 5 kali
Tersambung pada ”16:00:21”. test
ke 6. Berhasil 6 kali
Tersambung pada ”16:00:24”. test
ke 7. Berhasil 7 kali
Tersambung pada ”16:00:27”. test
ke 8. Berhasil 8 kali
Tersambung pada ”16:00:30”. test
ke 9. Berhasil 9 kali
Tersambung pada ”16:00:33”. test
ke 10. Berhasil 10 kali
Hasil pengujian ping server :
Berhasil    : 10 kali
Gagal       : 0 kali
Tindakan : Tidak mengirim SMS
```

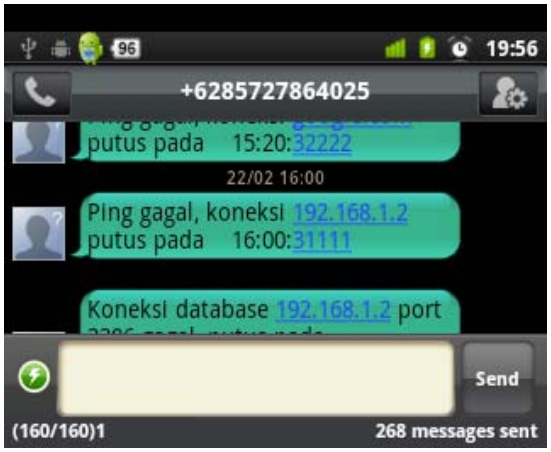
Gambar 6. Hasil monitoring konektifitas server dengan DNS dengan kondisi terhubung

Namun apabila PC A tidak dapat terhubung dengan PC B, hasil yang diperoleh dari monitoring ini dapat dilihat pada gambar 7.

```
Terputus pada ”16:00:04”. test ke
1. Gagal 1 kali
Terputus pada ”16:00:07”. test ke
2. Gagal 2 kali
Terputus pada ”16:00:10”. test ke
3. Gagal 3 kali
Terputus pada ”16:00:13”. test ke
4. Gagal 4 kali
Terputus pada ”16:00:16”. test ke
5. Gagal 5 kali
Terputus pada ”16:00:19”. test ke
6. Gagal 6 kali
Terputus pada ”16:00:22”. test ke
7. Gagal 7 kali
Terputus pada ”16:00:25”. test ke
8. Gagal 8 kali
Terputus pada ”16:00:28”. test ke
9. Gagal 9 kali
Terputus pada ”16:00:31”. test ke
10. Gagal 10 kali
Hasil pengujian ping server :
Berhasil    : 0 kali
Gagal       : 10 kali
Tindakan : Mengirim SMS
Written message with ID 51
```

Gambar 7. Hasil monitoring konektifitas server dengan DNS dengan kondisi tidak terhubung

Ketika melakukan monitoring, apabila dalam 10 kali percobaan akses gagal, maka sistem akan megirimkan SMS kepada admin seperti gambar 8.



Gambar 8. Pesan diterima admin

Fitur kedua adalah pengujian koneksi *database* server. Dalam topologi pengujian sederhana, PC A akan melakukan pengujian koneksi *database* terhadap PC C dengan cara memonitor port 3306 pada PC C menggunakan aplikasi Netcat. Bila PC A terhubung dengan *port database* PC C hasil monitoring yang diperoleh dapat dilihat pada gambar 9.

```
Tersambung dengan database pada
”16:00:04”. test ke 1. Berhasil 1
kali
Tersambung dengan database pada
”16:00:07”. test ke 2. Berhasil 2
kali
Tersambung dengan database pada
”16:00:10”. test ke 3. Berhasil 3
kali
Tersambung dengan database pada
”16:00:13”. test ke 4. Berhasil 4
kali
Tersambung dengan database pada
”16:00:16”. test ke 5. Berhasil 5
kali
Tersambung dengan database pada
”16:00:19”. test ke 6. Berhasil 6
kali
Tersambung dengan database pada
”16:00:22”. test ke 7. Berhasil 7
kali
Tersambung dengan database pada
”16:00:25”. test ke 8. Berhasil 8
kali
Tersambung dengan database pada
”16:00:28”. test ke 9. Berhasil 9
kali
Tersambung dengan database pada
”16:00:31”. test ke 10. Berhasil
10 kali
Hasil pengujian server database:
Berhasil    : 10 kali
Gagal       : 0 kali
Tindakan : Tidak mengirim SMS
```

Gambar 9. Hasil monitoring konektifitas *database* kondisi terhubung

Namun apabila PC A tidak dapat terhubung dengan *port database* PC C, yaitu bila dalam 10 kali akses gagal, sistem akan mengirimkan SMS kepada admin seperti gambar 10.

Gambar 10. Pesan diterima admin

Hasil monitoring akses *database* yang diperoleh dari percobaan ini dapat dilihat pada gambar 11.

```
Terputus dengan database pada
"16:10:04". test ke 1. Gagal 1
kali
Terputus dengan database pada
"16:10:07". test ke 2. Gagal 2
kali
Terputus dengan database pada
"16:10:10". test ke 3. Gagal 3
kali
Terputus dengan database pada
"16:10:13". test ke 4. Gagal 4
kali
Terputus dengan database pada
"16:10:16". test ke 5. Gagal 5
kali
Terputus dengan database pada
"16:10:19". test ke 6. Gagal 6
kali
Terputus dengan database pada
"16:10:22". test ke 7. Gagal
7 kali
Terputus dengan database pada
"16:10:25". test ke 8. Gagal 8
kali
Terputus dengan database pada
"16:10:28". test ke 9. Gagal 9
kali
Terputus dengan database pada
"16:10:31". test ke 10. Gagal 10
kali
Hasil pengujian server database :
Berhasil    : 0 kali
Gagal       : 10 kali
Tindakan : Mengirim SMS
Written message with ID 52
```

Gambar 11. Hasil monitoring konektifitas *database* kondisi tidak terhubung

Di dalam aplikasi gammu sudah terdapat *database* yang menyimpan semua pesan yang dikirim maupun diterima secara otomatis. Sehingga dapat dimanfaatkan menjadi SMS log untuk implementasi sistem *monitoring* jaringan menggunakan *SMS gateway*. Seperti yang terlihat pada gambar 12.

Nomor	Tanggal Pengiriman	Nomer tujuan	Pesan
1	2010-12-12 15:50:39	085726940720	Koneksi database 127.0.0.1 port 3307 gagal, putus pada "15:50:31"
2	2010-12-12 15:50:35	085726940720	Ping gagal, koneksi google.com putus pada "15:50:31"

Gambar 12. SMS log yang terkirim dalam database

6. Simpulan

Setelah dilakukan serangkaian percobaan maka didapatkan hasil sebagai berikut:

a. Sistem *monitoring* jaringan dibangun untuk dapat membantu administrator jaringan mengawasi kondisi koneksi server linux.
b. Dengan memperhatikan banyaknya SMS yang terkirim pada SMS log di *database*, administrator dapat mengambil tindakan untuk melakukan sesuatu terhadap jaringannya.

7. Daftar Pustaka

[1] Eric Cole, Ronald Krutz, & James W. Conley, 2005, "*Network Security Bible"*, Wiley Publishing, Inc. Indianapolis, United States.

[2] Ghazali, M. 2008, "*Topologi jaringan*", diakses dari http://www.scribd.com/doc/46315374/Topologi-Jaringan pada 12 november 2010 pukul 22.00 WIB.

[3] Lydia P, David T, Britt C. T. 2006, "*TCP/IP Tutorial and Technical Overview",* IBM International Technical Support Organization, New York, United States.

[4] Machtelt Garrels, 2008, "*Introduction to Linux*", diakses dari http://tldp.org/LDP/intro-linux/intro-linux.pdf, pada tanggal 10 januari 2011 pukul 11.00 WIB.

[5] Tito, 2004, "*Jaringan Komputer*" diakses dari http://blog.unikom.ac.id/v/L5/ pada 21 Oktober 2010 pukul 18.50 WIB.

[6] Tri Wicaksono, Mohamad. 2006, "*Pemrograman SMS Interaktif Berbasis Java",* Elex Media Komputindo, Jakarta.

[7] Vivek G. Gite. 1999-2002, "*Linux Shell basics*", diakses dari http://www.freeos.com/guides/lsst/ pada tanggal 23 Oktober 2010 pukul 18.50 WIB.