Dashboard Monitoring System Berbasis *Web* Sebagai Pemantau Layanan *liteBIG Instant Messenger*

Gigih Forda Nama[1], Abdul Munif Hanafi[2], Muhammad Bagus Nurfaif[2], M Tesar Sandikapura[3]
[1]Department of Informatics Engineering Lampung University, Indonesia
[2]Department of Electrical Engineering Lampung University, Indonesia
[3]CEO LiteBig, Indonesia

(E-mail: gigih@eng.unila.ac.id, am.hanafi@students.unila.ac.id, m.bagus66@gmail.com, tesar@gmail.com)

Abstrak -- Saat ini hampir semua pengguna ponsel pintar menggunakan layanan perpesanan instan sebagai media komunikasi dikarenakan layanan perpesanan instan lebih hemat biaya dengan hanya menggunakan jaringan *internet* dibandingkan layanan pesan singkat (*SMS*). Layanan yang diberikan harus dapat melayani pengguna dengan baik agar pesan yang dikirim oleh pengirim dapat diterima oleh penerima dengan cepat dan akurat. Layanan ini juga harus dijaga keandalannya untuk menjamin kualitas pelayanan dan untuk menghindari ketidaknyamanan pengguna. Sehingga, diperlukan adanya sistem pemantauan berupa perangkat lunak untuk pengawasan status layanan setiap saat dapat diakses dari manapun dan kapanpun. PT.Sandika Cahaya Mandiri memiliki produk layanan perpesanan instan dengan *brand name liteBIG Messenger*. Perusahaan ini memerlukan perangkat lunak untuk pemantauan layanan *liteBIG Messenger*. Dengan adanya perangkat lunak pemantauan layanan, petugas pemantauan dapat melihat secara *realtime* status layanan utama pada setiap komputer *server*, pemakaian sumber daya (*cpu*, *memory*, dan *harddisk*), dan statistik pengguna baru *liteBIG Messenger* melalui antarmuka *web*. Petugas pemantauan juga akan mendapat pemberitahuan ketika terjadi masalah pada layanan sehingga masalah dapat lebih dini diketahui dan *downtime* dapat dikurangi.

Kata Kunci: ***Monitoring, Service Monitoring, Instant Messaging, SSH Monitoring*.**

I. PENDAHULUAN

Aplikasi perpesanan instan adalah aplikasi yang wajib ada pada setiap ponsel pintar saat ini. Jenis aplikasi ini menjadi jalur komunikasi utama oleh sebagian besar pengguna ponsel pintar dikarenakan lebih hemat biaya dengan hanya menggunakan jaringan internet dibandingkan layanan pesan singkat (*SMS*). Aplikasi perpesanan instan harus dapat melayani pengguna dengan baik supaya pesan yang dikirim oleh pengirim dapat diterima oleh penerima dengan cepat dan akurat.

Di dalam sebuah sistem layanan perpesanan instan, terdapat beberapa komputer *server* untuk melayani berbagai macam permintaan dari *client*. Komputer-komputer *server* tersebut harus dijaga keandalannya untuk menjaga kualitas pelayanan kepada pengguna. Jika terjadi masalah pada komputer server, maka layanan akan terhenti dan menyebabkan ketidaknyamanan kepada pengguna. Petugas pemantauan tidak dapat secara terus menerus memantau status layanan dikarenakan keterbatasan manusia.

Oleh karena itu diperlukan perangkat lunak yang dapat membantu petugas pemantauan untuk memantau layanan secara *realtime* dan dapat menyampaikan pemberitahuan kepada petugas pemantauan ketika terjadi masalah pada layanan. Dengan adanya perangkat lunak ini, petugas pemantauan dapat mengetahui lebih dini masalah yang terjadi sehingga *downtime* dapat dikurangi.

II. TINJAUAN PUSTAKA

A. Server

Di dalam dunia komputasi, *server* adalah sebuah program komputer atau perangkat yang menyediakan fungsi untuk program atau perangkat lain, yang disebut "*client*". Arsitektur ini disebut model client-server, dan hasil komputasi didistribusikan di beberapa proses atau perangkat. *Server* dapat memiliki berbagai fungsi yang sering disebut sebagai "*service*" atau layanan, seperti berbagi data atau sumber daya di antara beberapa *client*, atau melakukan komputasi untuk *client*. *Service* adalah program-program yang umumnya berjalan latar belakang dan melayani tugas-tugas tertentu yang diperlukan oleh program lain. Sebuah *server* tunggal dapat melayani beberapa *client*, dan *client* tunggal dapat menggunakan beberapa *server*. *Server* yang biasanya digunakan adalah *database servers, file servers, mail servers, print servers, web servers, game servers*, dan *application servers* [1].

B. Server Aplikasi Mobile

Sebuah *server* aplikasi *mobile* adalah *mobile middleware* yang membuat sistem *back-end* dapat diakses aplikasi *mobile* untuk mendukung pengembangan aplikasi *mobile*. Sama seperti *web server* yang menyimpan, memproses, dan memberikan halaman web untuk klien, *server* aplikasi mobile menjembatani kesenjangan dari infrastruktur yang ada untuk perangkat *mobile*. Meskipun sebagian besar standar infrastruktur dirancang untuk terhubung ke berbagai jenis vendor, produk, atau teknologi, tetapi sebagian besar perusahaan mengalami kesulitan menghubungkan sistem *back-end* untuk aplikasi *mobile*, karena perangkat *mobile* memiliki beberapa tantangan teknologi seperti berikut:

- sumber daya yang terbatas - perangkat *mobile* memiliki sumber listrik dan *bandwidth* yang terbatas
- konektivitas intermiten - cakupan layanan seluler dan *WiFi* sering tidak terus menerus
- sulit untuk mengamankannya - mobilitas dan *BYOD (Bring Your Own Device)* membuat sulit untuk mengamankan perangkat *mobile* [2].

C. Secure Shell (SSH)

Secure Shell adalah program yang melakukan *logging* terhadap komputer lain dalam jaringan, mengeksekusi perintah lewat mesin secara *remote*, dan memindahkan file dari satu mesin ke mesin lainnya. *SSH* menggunakan kriptografi *public key* untuk mengenkripsi komunikasi-komunikasi antara dua *host*. Ada banyak program-program *SSH* seperti *sshd*, *slogin*, *scp*, dan lain-lain [3].

D. Aplikasi Web

Aplikasi *web* adalah aplikasi perangkat lunak *client-server* yang dapat dijalankan oleh pengguna melalui *web browser*. Aplikasi *web* sangat populer dikarenakan pengguna hanya membutuhkan *web browser* untuk mengaksesnya tanpa harus memasang perangkat lunak tambahan dan mendukung kompatibilitas *cross-platform*. Aplikasi *web* yang umum dipakai adalah *webmail*, jual beli *online*, lelang *online*, layanan perpesanan instan, dan lain-lainnya.

E. Bahasa Pemrograman Python

Python merupakan salah satu bahasa pemrograman tingkat tinggi dengan tipe bahasa *interpreted* karena program-program *Python* langsung dieksekusi oleh *interpreter* tanpa harus melalui tahap kompilasi [4]. *Python* dapat dijalankan dengan dua cara, yakni:

1. Mode *command-line*
2. Mode *script*

Python memiliki banyak *library* yang dapat digunakan oleh *programmer* pada setiap saat. Ini dapat membuat *programmer* tidak harus menulis setiap program dari awal, *programmer* dapat menggunakan hal-hal yang sudah ada di library untuk melakukan banyak hal yang dimungkinkan. Inilah yang membuat *Python* menjadi salah satu bahasa pemrograman yang begitu kuat [5].

F. CodeIgniter Web Framework

CodeIgniter adalah *framework* yang bersifat *open source* untuk membuat aplikasi *web* atau digunakan untuk membangun *website* dinamis dengan bahasa perograman *PHP*. Tujuannya adalah untuk mempercepat pengerjaan

proyek-proyek daripada harus menulis kode dari awal. Di dalam *CodeIgniter* sudah terdapat satu set pustaka untuk tugas-tugas yang biasa diperlukan, serta antarmuka yang sederhana dan struktur logis untuk mengakses pustaka tersebut [6]. *Codelgniter* menggunakan model pengembangan populer yaitu *Model-View-Controller*. *View* dan *Controller* adalah bagian penting dari *framework Codelgniter*, dan *Model* adalah opsional. *Codelgniter* memiliki performa yang lebih cepat dibandingkan dengan *framework PHP* yang lainnya [6].

III. KEBUTUHAN SISTEM

Perangkat lunak yang akan digunakan untuk pemantauan status layanan pada perusahaan ini memiliki kebutuhan sebagai berikut:

Kebutuhan fungsional:
1. Dapat mengetahui nilai persentase pemakaian sumber daya *cpu*, *memory*, dan *harddisk.*
2. Dapat mengetahui status sebuah layanan.
3. Dapat mengetahui statistik pngguna baru hari ini, kemarin, minggu ini, bulan ini, dan total pengguna.
4. Dapat mengetahui status layanan perpesanan dan mendapatkan nilai *delay*-nya.
5. Dapat mengeksekusi perintah tertentu ketika terjadi masalah atau terjadi kegagalan sistem.
6. Dapat mengeksekusi perintah tertentu jika pemakaian sumber daya melebihi ambang batas yang ditentukan.

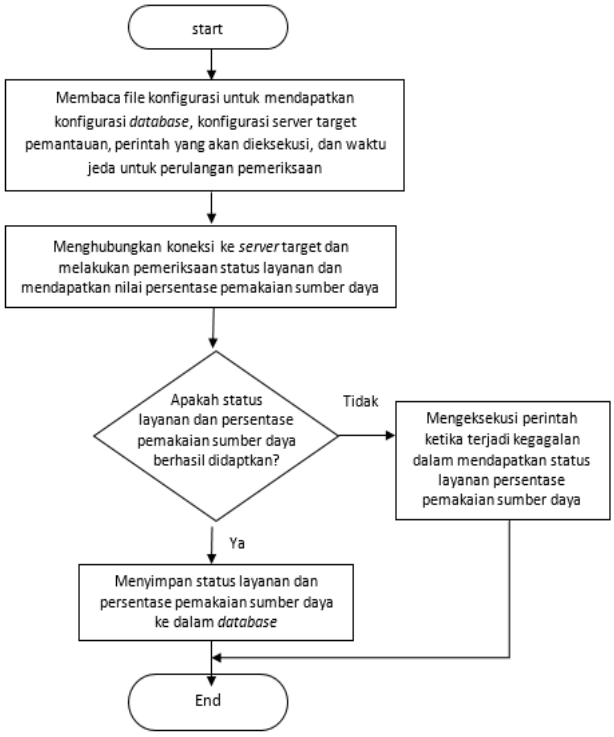
Kebutuhan non-fungsional:
1. Status terkini hasil pemantauan dapat dilihat oleh petugas melalui antarmuka *web*.
2. Status layanan dan nilai persentase pemakaian suber daya diperoleh dengan mengeksekusi perintah melalui *SSH* tanpa *password* (*passwordless SSH*).
3. Memiliki *file* konfigurasi yang berisi:
   - pengaturan *database*
   - rincian target *server* yang akan dipantau
   - perintah untuk mendapatkan nilai persentase pemakaian sumber daya
   - perintah untuk mendapatkan status sebuah layanan
   - nilai ambang batas pemakaian sumber daya dan perintah yang akan dieksekusi ketika pemakaian sumber daya melebihi ambang batas
   - perintah yang akan dieksekusi ketika terjadi masalah pada sebuah layanan
4. Bahasa pemrograman menggunakan bahasa *Python* dan *PHP*.
5. *Database* untuk penyimpanan mengunakan *MySQL*.

IV. PERANCANGAN

Perancangan sistem untuk pembuatan program ini dilakukan untuk memberikan gambaran tentang sistem yang akan dibuat.

A. Diagram Alir

Perancangan diagram alir untuk sistem ini adalah sebagai berikut:

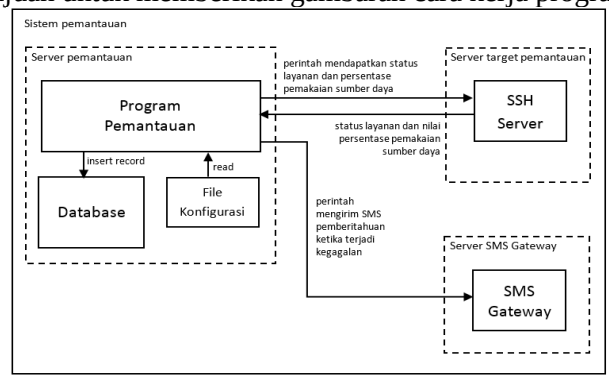


Gambar 1. Diagram Alir Sistem

Diagram alir sistem pada gambar 1 di atas menggambarkan alur program yang dibuat, yaitu pertama program akan membaca *file* konfigurasi yang berisi pengaturan database, rincian *server* target pemantauan, perintah-perintah yang akan dieksekusi untuk mendapatkan status layanan dan nilai persentase pemakaian sumber daya, serta perintah yang akan dieksekusi ketika terjadi kegagalan ketika mendapatkan status layanan dan pemakaian sumber daya. Kemudian program akan menghubungkan koneksi ke *server* target pemantauan dan mulai melakukan pemantauan. Jika status layanan dan persentase pemakaian sumber daya berhasil didapatkan maka program akan menyimpan data tersebut ke dalam database, dan jika tidak berhasil maka program akan mengeksekusi perintah yang sudah diatur di dalam file konfigurasi.

B. Diagram Konseptual

Diagram konseptual ini bertujuan untuk memberikan gambaran cara kerja program, yaitu sebagai berikut:



Gambar 2. Perancangan Konseptual

Gambar 2 di atas menggambarkan perancangan konseptual dimana terdapat tiga buah bagian utama di dalam sistem yaitu *server* pemantauan, *server* target pemantauan, dan *server SMS Gateway*. Di dalam *server* pemantauan terdapat program utama untuk melakukan pemantauan, file konfigurasi, dan *database*. Sebelum melakukan pemantauan,

program pemantauan membaca file konfigurasi terlebih dahulu kemudian mengeksekusi perintah untuk mendapatkan status layanan dan persentase pemakaian sumber daya pada *server* target pemantauan melalui protokol *SSH*. Sebelumnya, di dalam *server* target pemantauan harus sudah terpasang layanan *SSH* dan berjalan. Jika status layanan dan nilai persentase pemakaian sumber daya berhasil didapatkan, data tersebut kemudian disimpan di dalam database. Dan jika status layanan dan nilai persentase pemakaian sumber daya gagal didapatkan, maka program akan mengeksekusi perintah yang sudah diatur di dalam file konfigurasi. Pada pembuatan program ini perintah yang akan dieksekusi adalah perintah untuk melakukan pengiriman SMS pemberitahuan melalui server SMS gateway.

V. APLIKASI PEMANTAUAN

Pembuatan perangat lunak pemantauan menggunakan bahasa pemrograman *Python*. Pustaka tambahan yang digunakan adalah *os* dan *sys* untuk mengeksekusi perintah sistem operasi, *time* dan *datetime* untuk fungsi waktu, *pymysql* untuk koneksi ke *database MySQL*, *stomp* untuk koneksi ke *server messaging*, *random* untuk membangkitkan angka acak, *requests* untuk mengirim permintaan ke *server HTTP*, *ConfigParser* untuk membaca *file* konfigurasi, dan *multiprocessing* untuk membuat banyak proses yang digunakan untuk memisahkan setiap eksekusi perintah pemeriksaan menjadi proses sendiri.

Untuk mendapatkan nilai persentase pemakaian sumber daya dilakukan dengan cara mengeksekusi perintah melalui *SSH* tanpa *password* (*passwordless*). Syarat untuk melakukan *SSH* tanpa *password* ini adalah harus memasang identitas pengguna dari *server* pemantau terlebih dahulu di dalam *server* yang akan dipantau. Dengan menggunakan *SSH* tanpa *password* ini pengguna dari *server* pemantau dapat mengeksekusi perintah di dalam *server* yang akan dipantau tanpa harus melakukan *log in* terlebih dahulu.

Untuk melakukan perintah *SSH* digunakan pustaka untuk mengeksekusi perintah sistem operasi yang kemudian didapatkan pesan kembalian dari perintah tersebut untuk diolah oleh program. Perintah yang akan dieksekusi dapat diatur pada file konfigurasi, sehingga dapat disesuaikan dengan sistem operasi yang dipakai pada *server* target pemantauan.

Pada saat program dijalankan, dilakukan pemeriksaan *file* konfigurasi. Jika terdapat kesalahan pada *file* konfigurasi, program akan dihentikan dan menampilkan pesan kesalahan. Jika tidak terdapat kesalahan pada *file* konfigurasi, program akan berjalan dan membuat proses-proses baru sesuai dengan jumlah sumber daya dan layanan yang dipantau. Nilai persentase pemakaian sumber daya disimpan ke dalam *database MySQL*, data bertambah setiap interval tertentu sesuai dengan yang diatur di dalam *file* konfigurasi. Status layanan juga disimpan ke dalam *database MySQL* dan diperbaharui setiap interval tertentu sesuai dengan yang diatur di dalam *file* konfigurasi. Sama halnya juga dengan status layanan perpesanan dari *server* perpesanan yang dipantau.

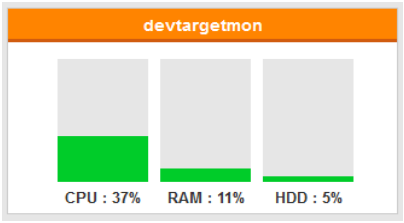
A. Antarmuka Web

Untuk memudahkan petugas pemantauan maka dibuat antarmuka *web* untuk melihat status terkini dari layanan yang dipantau dan pemakaian sumber daya. Pembuatan antarmuka *web* menggunakan bahasa pemrograman *PHP* dengan *framework CodeIgniter*. Terdapat dua *controller* utama yaitu *controller Main* yang digunakan untuk menampilkan halaman *dashboard* dan *controller Get* yang digunakan untuk mendapatkan data yang diperoleh dari *database* untuk ditampilkan di *dashboard web*. Di dalam *controller Get* terdapat tiga fungsi yaitu:

- *rmon($servername)*

Fungsi ini adalah untuk mendapatkan nilai persentase pemakaian sumber daya. Fungsi ini dapat diakses melalui *URL http://<hostname> /dashboard/get/rmon/<nama_server>* dengan data kembalian berupa *JSON* yang akan diolah oleh *javascript* di halaman *dashboard* dan kemudian ditampilkan nilai dan visualisasinya. Contoh data *JSON* yang diperoleh

```
{
  "cpu": "37.00",
  "mem": "11.36",
  "hdd": "5.00"
}
```

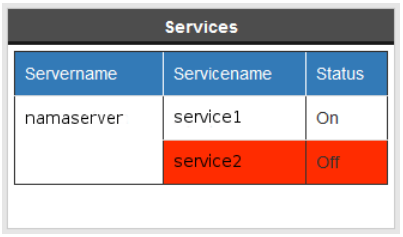


Gambar 3. Hasil visualisasi pemakaian sumber daya dari data JSON yang didapat

- *svcstats()*

Fungsi ini adalah untuk mendapatkan semua status layanan yang dipantau. Fungsi ini dapat diakses melalui *URL http://<hostname> /dashboard/get/svcstats* dengan data kembalian berupa *JSON* yang akan diolah oleh *javascript* dan kemudian ditampilkan di halaman dashboard. Contoh data *JSON* yang diperoleh


```
{
  "namaserver": {
    "service1": {
      "status": "On",
      "updatetime": "2016-04-14 05:33:14"
    },
    "service2": {
      "status": "Off",
      "updatetime": "2016-04-14 05:33:15"
    }
  }
}
```

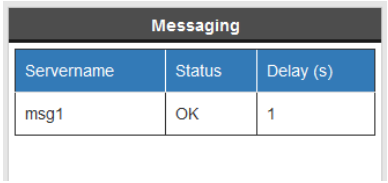
Gambar 4. Tampilan status service (layanan)

- *messaging()*

Fungsi ini adalah untuk mendapatkan status layanan perpesanan dan waktu *delay*-nya. Fungsi ini dapat diakses melalui *URL http://<hostname> /dashboard/get/svcstats* dengan data kembalian berupa *JSON* yang akan diolah oleh *javascript* dan kemudian ditampilkan di halaman *dashboard web*.

Contoh data *JSON* yang diperoleh


```
{
  "msg1": {
    "status": "OK",
    "updatetime": "2016-04-14 19:10:48",
    "delay": 1,
    "expired": "30"
  }
}
```

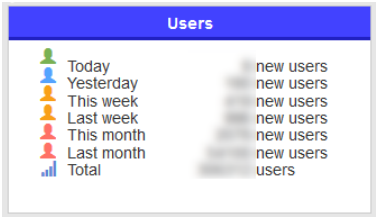


Gambar 5. Tampilan status layanan perpesanan

Untuk medapatkan statistik pengguna baru, digunakan *API* khusus di *server* eksternal yang diakses melalui protokol *HTTP* dengan parameter tambahan tanggal pertama dan tanggal ke dua sesuai dengan rentang yang dikehendaki dengan data kembalian berformat *JSON*.

data *JSON* yang diperoleh

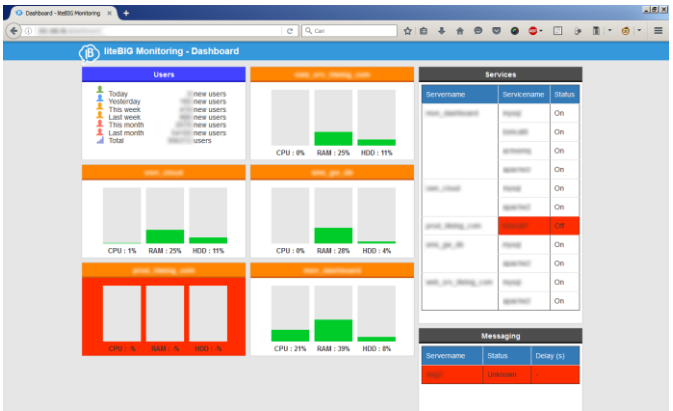
```
{
   count: "2083",
   date1: "2016-04-01",
   date2: "2016-04-30"
}
```



Gambar *6* . Tampilan statistik pengguna baru

B. Pengujian

Pengujian perangkat lunak ini dilakukan di dalam sebuah *server* yang sudah disiapkan. Terdapat beberapa *server* target yang dipantau dengan beberapa *service* di dialamnya, dan terdapat sebuah *server* perpesanan. Rincian *server* yang akan dipantau dimasukkan ke dalam *file* konfigurasi dan selanjutya program dijalankan. Hasil dari pemantauan ditampilkan pada halaman *web dashboard* seperti pada gambar di bawah ini



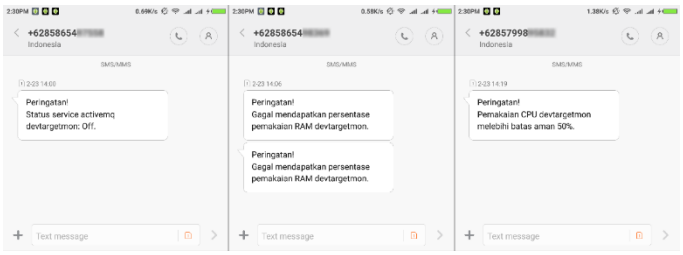
Gambar 7 . Halaman utama aplikasi pemantauan

Gambar di atas merupakan salah satu tampilan pada saat pengujian. Dimana, terdapat 5 buah *server* yang sedang dipantau. Pada *server* nomor 1 pemakaian *CPU* sebesar 0%, *RAM* sebesar 23%, dan *HDD* sebesar 11%. Pada *server* nomor 2 pemakaian *CPU* sebesar 1%, *RAM* sebesar 25%, dan *HDD* sebesar 11%. Pada *server* nomor 3 pemakaian *CPU* sebesar 0%, *RAM* sebesar 28%, dan *HDD* sebesar 4%. Pada *server* nomor 4 nilai persentase pemakaian sumber daya

tidak diketahui, hal ini dikarenakan perangkat lunak pemantauan gagal terhubung ke *server* nomor 4 sehingga tidak dapat mengetahui persentase pemakaian sumber daya saat itu juga. Pada *server* nomor 5 pemakaian *CPU* sebesar 21%, *RAM* sebesar 39%, dan *HDD* sebesar 8%.

Terlihat pada gambar di atas, *server* keempat pada bagian pemantauan sumber daya memiliki latar belakang berwarna merah yang berarti *server* tersebut tidak dapat diakses. Pada bagian pemantauan *service* di *server* ketiga terdapat *service* yang berstatus *Off*. Dan pada bagian pemantauan layanan perpesanan terdapat juga masalah pada *server* yang sedang dipanatau. Ketika terjadi masalah seperti ini, petugas pemantau akan melakukan pemeriksaan lebih lanjut terhadap *server* yang bermasalah tersebut dan melakukan tindakan yang diperlukan sehingga layanan dapat berjalan kembali.

Pengujian *alert* dilakukan dengan berbagai perlakuan. Pada pengujian status *service*, dilakukan dengan menonaktifkan salah satu *service*. Pada pengujian *alert* mendapatkan nilai persentase pemakaian sumber daya, dilakukan dengan mengubah nilai ambang batas. Perintah yang dieksekusi ketika terjadi masalah adalah perintah untuk mengakses *URL* untuk mengirim *SMS* melalui *SMS Gateway* dengan parameter tambahan berupa *username*, *key*, nomor ponsel tujuan, dan isi pesan. Contoh *SMS* pemberitahuan ketika terjadi masalah adalah seperti gambar di bawah ini



Gambar 8. SMS pemberitahuan ketika terjadi masalah

Gambar 8 di atas adalah hasil dari pengujian alert melalui *SMS*. Gambar nomor 1 adalah *SMS* pemberitahuan ketika salah satu *service* berstatus *Off*. *SMS* ini dikirimkan setelah dilakukan penonaktifan salah satu *service* yang sedang dipantau secara sengaja. Pada gambar nomor 2 adalah *SMS* pemberitahuan ketika perangkat lunak pemantau gagal terhubung ke *server* target pemantauan untuk mendapatkan nilai persentase pemakaian sumber daya *RAM*. *SMS* ini dikirimkan setelah koneksi ke *server* yang dipantau diputus secara sengaja. Dan pada gambar 3 adalah *SMS* peringatan ketika pemakaian salah satu sumber daya yaitu *CPU* melebihi *threshold* yang ditentukan pada *file* konfigurasi. Pada pengujian ini nilai *threshold* adalah 50, sehingga ketika pemakaian *CPU* melabihi 50% maka *SMS* peringatan akan segera dikirimkan. Waktu rata-rata pengiriman *SMS* pemberitahuan untuk sampai ke petugas adalah kurang dari 30 detik.

VI. KESIMPULAN

Kesimpulan yang diperoleh dari implementasi aplikasi pemantauan ini adalah sebagai berikut:

1. Dengan adanya perangkat lunak pemantauan layanan, petugas dapat mengetahui status terkini layanan yang sedang dipantau, pemakaian sumber daya (cpu, memory, dan harddisk), dan statistik pengguna baru liteBIG Messenger melalui antarmuka web.
2. Petugas pemantauan menerima SMS pemberitahuan ketika terjadi kegagalan dalam waktu kurang dari 30 detik.
3. Pemberitahuan oleh perangkat lunak pemantauan dapat memberikan informasi lebih dini kepada petugas ketika terjadi masalah pada layanan.

DAFTAR PUSTAKA

[1] Microsoft Official Academic Course. 2010. *Windows Server Administration Fundamentals.* New York: John Wiley and Sons.

[2] http://appdevelopermagazine.com/ 2104/2014/11/19/Why-Mobile-App-Development-Requires-More-than-an-SOA), diakses 12 April 2015

[3] Stiawan, Deris. 2005. *Sistem Keamanan Komputer*. Jakarta: PT. Elex Media Komputindo.

[4] Utami, E., Raharjo, S. 2004. *Logika Algoritma dan Implementasinya dalam Bahasa Python di GNU/Linux*. Yogyakarta: Andi.

[5] Mount, S., Shuttleworth, J., Winder, R. 2008. *Python for Rookies: A First Course in Programming*. London: Thomson Learning (EMEA) Ltd.

[6] Sulaiman, H.A., Othman, M.A., Othman, M.F.I, Rahim, Y.A., Pee, N.M. 2014. *Advanced Computer and Communication Engineering Technology*. Malaysia: Springer.