LONTAR KOMPUTER VOL. 7, NO.2, AGUSTUS 2016 **p-ISSN 2088-1541**
DOI: 10.24843/LKJITI.2016.v07.i02.p05 **e-ISSN 2541-5832**

Perancangan *Network Monitoring Tools* Menggunakan *Autonomous Agent Java*

Khurniawan E S[a1], L.Ahmad S. Irfan A[a2] ,I B K Widiartha[b3]

[a]Jurusan Teknik Elektro Fakultas Teknik Universitas Mataram
Jln. Majapahit No.62 Mataram Nusa Tenggara Barat Kode Pos: 83125
[1,2]Telp. (0370) 636087; 636126; ext 128 Fax (0370) 636087

[b]Program Studi Teknik Informatika Universitas Mataram
Jln. Majapahit No.62 Mataram Nusa Tenggara Barat Kode Pos: 83125
[3]Telp. (0370) 636087; 636126; ext 128 Fax (0370) 636087

Abstrak

Tugas pengelolaan jaringan yang dilakukan administrator jaringan diantaranya yaitu pengumpulan informasi resource jaringan yang tersedia. Teknologi SNMP (Simple Network Management Protocol) memberikan fleksibilitas bagi administrator jaringan dalam mengatur network secara keseluruhan dari satu lokasi. Aplikasi Network Monitoring Tools berbasis Agent JAVA terdiri dari Master agent yang bertugas untuk melakukan management Request agent serta akses database. Request agent yang bertugas untuk melakukan pemantauan server yang mengimplementasi library SNMP4j dengan sistem multi-agent. Disisi interface, aplikasi Network Monitoring Tools menggunakan media web sebagai interface administrator sehingga dapat digunakan darimana saja dan kapan saja. Hasil dari penelitian ini memperlihatkan bahwa aplikasi yang dibuat bekerja sebagai Network Monitoring Tools mampu bekerja dengan persen error pada kisaran 0-18%. Selain itu Aplikasi ini menghasilkan tren pembacaan data server lebih stabil dan cepat dibandingkan dengan aplikasi Cacti. Hal ini didukung oleh kemampuan Request Agent yang mampu merespon tingkat beban kerja server yang di pantau.

***Kata kunci** : Agent, JAVA, SNMP, Cacti, Network monitoring.*

Abstract

Network management tasks are performed by the Network Administrator such as gathering available network resources information. The SNMP (Simple Network Management Protocol) Technology provides flexibility for network administrators in managing the overall network from a single location. Agent based Network Monitoring Tools JAVA Application consists of Master agent whose job is to perform as well as the management agent Request and database access. The Request agent in charge for monitoring servers that implement SNMP4j library with multi - agent systems. For interface, Network Monitoring Tools application using web media as an administrator interface that can be used from anywhere and at anytime. The results of this study showed that the application as a Network Monitoring Tools are able to work with the percent error in the range of 0-18 % . Besides these applications generate trend data readout server more stable and faster than the application Cacti do. This is supported by the ability of the Request Agent to respond the level of server workloads

***Keywords** : Agent, JAVA, SNMP, Cacti, Network monitoring.*

1. Pendahuluan

Tugas pengelolaan jaringan yang dilakukan *administrator* jaringan memiliki sejumlah kesulitan, diantaranya yaitu pengumpulan informasi *resource* jaringan yang tersedia serta melakukan pemantauan terhadap *server* dan *router* yang beroperasi pada jaringan. Manajemen sistem jaringan komputer berbasiskan pada teknologi SNMP (*Simple Network Management Protocol*)

yang memberikan fleksibilitas bagi *administrator* jaringan dalam mengatur *network* secara keseluruhan dari satu lokasi [1]. Banyak aplikasi 3rd *party* seperti cacti, MRTG, monit, munin,

nagios, zenos dan zabix yang menyediakan banyak fitur dalam manajemen sistem jaringan komputer menggunakan protokol SNM, dan diakses melalui web browser [2]. Namun aplikasi-aplikasi tersebut cukup sulit dikustomisasi menyesuaikan dengan kebutuhan instansi pengguna. Dibutuhkan sebuah aplikasi mandiri yang sederhana agar mudah disesuaikan dengan kebutuhan dan memiliki fitur-fitur *monitoring* seperti pada aplikasi 3rd *party* pada umumnya. Untuk menjawab permasalahan tersebut maka dibuatlah sebuah aplikasi *Network Monitoring Tools* (NMT) berbasis *Agent* JAVA. *Agent* JAVA mampu bekerja secara berulang-ulang (*Autonomous*) dan merespon sesuai dengan parameter yang diterima. Penggunaan *Agent* JAVA sebagai pemantau *server* yang mampu merespon kondisi pembebanan *server* dalam bentuk interval pengambilan datanya (*request scheduling*) diharapkan dapat menjadi solusi untuk aplikasi mandiri *Network Monitoring Tools* pada jaringan Fakultas Teknik Universitas Mataram.

2. Landasan Teori

2.1. SNMP

Secara sederhana, SNMP merupakan sebuah protokol yang didesain untuk memberikan kemampuan kepada pemakai untuk mengelola jaringan komputernya dari jarah jauh atau remote[3]. Adanya SNMP memungkinkan manajemen jaringan yang tersentralisasi, kuat, dan kompatibel pada semua platform. Tujuan utama dari protokol SNMP hanya pada satu tujuan yaitu melakukan *remote* manajemen dari peralatan [4]. Sebuah jaringan yang dapat di-*manage* menggunakan SNMP pada dasarnya memiliki tiga komponen, yaitu *Managed Device, Agent SNMP,* dan *Network-management System* [4]. Sebuah *managed device* adalah sebuah *node* (dapat berupa *server, switch, router*) di jaringan yang berisi *Agent SNMP* yang berada di jaringan yang dapat di *manage*. *Agent* SNMP adalah sebuah modul *software network management* yang berada di dalam *managed device*. *Agent* ini mengetahui tentang informasi manajemen dan dalam menterjemahkan ke informasi yang kompatibel dengan SNMP. Pada sistem operasi linux telah disediakan aplikasi snmpd sebagai *agent* SNMP. Aplikasi NMS menjalankan aplikasi yang dapat memonitor dan mengontrol *managed device*. NMS memberikan *resource memory* dan *prosesor* yang dibutuhkan untuk manajemen *network*.

2.2. Agent Java

Software Agent (selanjutnya di sebut *Agent* saja) adalah entitas perangkat lunak yang didedikasikan untuk tujuan tertentu. *Agent* bisa memiliki ide sendiri mengenai bagaimana menyelesaikan suatu pekerjaan tertentu atau agenda tersendiri. Karakteristik dari *Agent* [5]:

a. *Autonomy*: komputer umumnya hanya berespon pada manipulasi langsung. Kontras dengan *Agent* perangkat lunak yang mengamati lingkungannya dan bisa melakukan tindakan otonom.
b. *Reaktif*: suatu software Agent berespon dalam waktu yang bermacam-macam untuk merubah dalam lingkungannya
c. *Goal driven*: suatu Agent bisa menerima *Request* tingkat tinggi yang menentukan tujuan dari *user* manusia (atau *Agent* lainnya) dan memutuskan dimana dan bagaimana untuk menjawab *Request* tersebut.

Agent yang tidak berpindah ke *host* lain disebut *stationary Agent*. *Multi agent system* adalah sebuah *system* yang memungkinkan sejumlah *agent* (*multi agent*) bekerja sama untuk menyelesaikan suatu masalah yang tidak dapat dikerjakan oleh individu *agent* [5]. Pada *multi agent system*, setiap *agent* bekerja secara *multithreading* diatas *runtime* JAVA sehingga memberikan impresi bahwa *agent* mampu bekerja secara *paralel independent*.

2.3. JADE

JADE (*Java Agent DEvelopment Framework*) merupakan sebuah kerangka kerja perangkat lunak yang diimplementasikan sepenuhnya dalam bahasa *Java* [6]. JADE memudahkan implementasi dari *Multi-Agent sistem* (MAS) melalui *middleware* yang bekerja sesuai spesifikasi FIPA (*Foundation for Intelligent Physical Agent*). Karena sepenuhnya diimplementasikan dalam bahasa pemrograman *Java*, maka JADE juga mendapatkan seluruh keuntungan dari bahasa pemrograman tersebut, termasuk ketidak-tergantungan pada arsitektur *platform*. *Agent platform*

pada JADE dapat didistribusikan di beberapa mesin yang berbeda, dan tidak perlu menggunakan sistem operasi yang sama.

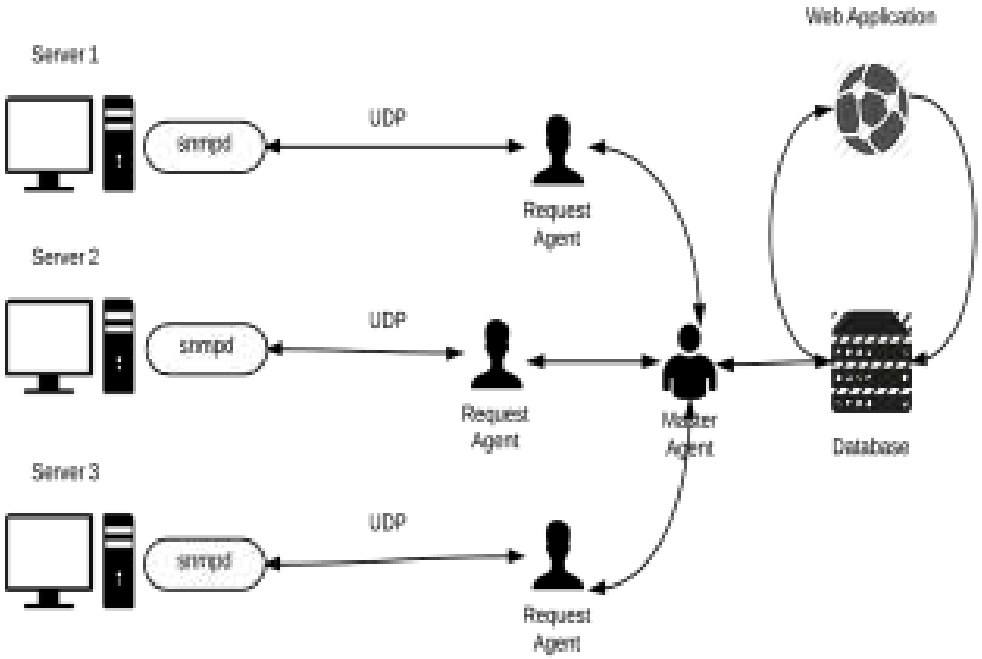
3. Metode Penelitian

3.1. Analisis Kebutuhan

Sistem *monitoring* yang saat ini berjalan banyak menggunakan *tools-tools* buatan luar negeri yang memanfaatkan protokol SNMP sebagai media *monitoring* pada suatu *server*. Aplikasi yang baru tetap menggunakan protokol SNMP sebagai media *monitoring* agar tidak terjadi perubahan langsung pada konfigurasi *server* dengan adanya penggantian aplikasi *monitoring*. *Network Monitoring Tools* yang akan dibangun adalah pemanfaatan dari teknologi *Agent JAVA,* yang mana untuk proses *monitoring* dalam satu *server* ditangani oleh sebuah *Agent.* Semakin banyak *host* yang perlu dipantau maka *Agent* yang diperlukan juga akan semakin banyak dan membutuhkan spesifikasi komputer yang cukup tinggi. *Interval Request Data* akan diatur oleh *Agent* dengan memantau tingkatan *traffic* jaringan dan beban kerja *host*. Ini artinya *Agent* yang akan melakukan *Request* data akan secara bergiliran melakukan pengambilan data, sehingga *traffic* jaringan lebih stabil tanpa mengurangi performa aplikasi *monitoring*.

3.2. Perancangan Sistem

Perangkat lunak yang akan dibangun adalah sebuah *tools* untuk memantau jaringan atau *Network Monitoring Tools*. Pemantauan jaringan dilakukan dengan cara memanfaatkan protokol SNMP untuk mengakses MIB (*Management Information Bases*) tiap *host* dan mendapatkan data sesuai OID (*Object ID*) yang dikirimkan oleh aplikasi *daemon* snmpd. Proses permintaan data *resource* menggunakan *library* JAVA SNMP4j dan menghasilkan *data set* dari *resource server* yang dipantau. *Data set* ini kemudian di simpan kedalam *database* dan ditampilkan pada aplikasi *web* yang *di* akses oleh *client* secara *realtime* sesuai data yang diterima.



Gambar 1. Ilustrasi Sistem Monitoring

Berikut penjelasan alur proses monitoring jaringan :

a. Dengan menggunakan *interface web*, *admin* mendaftarkan semua *server* yang ingin dipantau kedalam Sistem
b. Perangkat lunak yang akan dibangun adalah sebuah tools untuk memantau jaringan atau *Network Monitoring Tools*. Pemantauan jaringan dilakukan dengan cara memanfaatkan protokol SNMP untuk mengakses MIB (*Management Information Bases*) dan disimpan pada *database komputer monitoring*. Pendaftaran terdiri dari IP *address*, *port* SNMP, dan *community Key* untuk mengakses snmpd pada *server* tersebut.
c. Aplikasi mencetak 2 jenis *Agent* yaitu *Master Agent* dan *Request Agent*. *Master Agent* adalah *Agent* yang bertugas untuk mencetak *Request Agent* sesuai dengan *data host* yang terdapat pada *database*, menerima data *resource* jaringan dari *Request Agent* dan meng-*entry* data tersebut ke *database* aplikasi. *Request Agent* adalah *Agent* yang akan melakukan *Request data* ke *server* yang ingin dipantau, melakukan pengolahan data mengirimkan hasilnya ke *Master Agent*.
d. *Request Agent* akan melakukan *Request resource data* yang terdiri dari *bandwidth traffic*, *memory usage*, CPU *usage*, dan *disk usage* ke *server* secara bergantian dengan interval waktu berdasarkan tingkat beban kerja dan *traffic server*. Interval waktu ini diperoleh dari kategori tingkat *traffic* data dan penggunaan CPU serta RAM.
e. Pada *Web Application*, sistem akan mengakses *database* secara *realtime* untuk mendapatkan informasi *resource* jaringan yang telah dikumpulkan oleh *Master Agent*.

3.3. Programming

Berikut beberapa prosedur yang digunakan dalam pembuatan program *Network Monitoring* JAVA :

a. Prosedur utilitas sistem
Prosedur ini berfungsi sebagai pendukung aplikasi *Agent*. Terdiri dari koneksi *database*, akses file konfigurasi dan penyedia *logging error*.

b. Prosedur *Agent*
Prosedur ini berfungsi sebagai prosedur utama *agent*. Pada prosedur *agent* terdiri dari 2 *class* yaitu *class master agent* dan *class request agent.*

c. Prosedur komunikasi *Agent*
Prosedur ini digunakan oleh *Agent* untuk berkomunikasi dan bertukar data antar *Agent*.

d. Prosedur pemantauan *server*
Prosedur ini digunakan oleh *Agent* untuk melakukan pemantauan *server.* Tediri dari pembentukan koneksi *protocol* UDP, pengaksesan *protocol* SNMP dan proses-proses pengolahan data yang diterima oleh *Agent*

3.4. Pengujian

Dalam pembuatan sistem, pengujian ditujukan untuk mengetahui kinerja sistem dalam melakukan proses pengumpulan data *resource server*. Tahapan pengujian yang dilakukan antara lain :

a. Pengujian pembacaan data protokol SNMP
Protokol SNMP diakses dengan memanfaatkan *library* tambahan SNMP4j dengan *output* berupa sebuah *set data* yang siap disimpan ke *database.* Pengujian dilakukan dengan memeriksa apakah *library* tersebut dapat memberikan respon *output set data* sesuai *input* OIDnya.
b. Pengujian perbandingan hasil pemantauan dengan aplikasi Cacti

Pengujian perbandingan hasil pemantauan dengan aplikasi Cacti berfungsi untuk mengetahui kinerja *Network Monitoring Tools* dibandingkan dengan kinerja cacti sebagai aplikasi yang telah banyak digunakan.

4. Hasil dan Pembahasan

4.1. Pengujian pembacaan data protokol SNMP

Perbandingan memanfaatkan aplikasi *top*, *bmon* dan *df sebagai referensi daa real* dihasilkan data sebagai berikut :

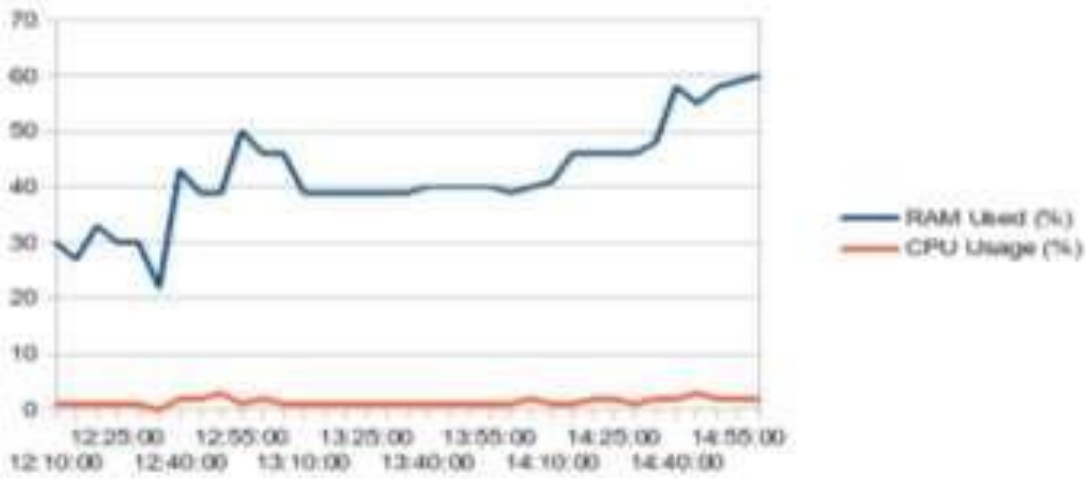
Tabel 1. Perbandingan hasil pemantauan system

No	Item	Data real	Data pemantauan	%error
1	Processor	78.7 %	78 %	3.74 %
2	RAM Used	363 MB	321 MB	11.57 %
3	Traffic IN	17 KBps	14 KBps	17.64 %
4	Traffic OUT	47 KBps	42 KBps	10.63 %
5	Disk Used	10.3 GB	10.3 GB	0 %
6	Disk Free	130 GB	130 GB	0 %

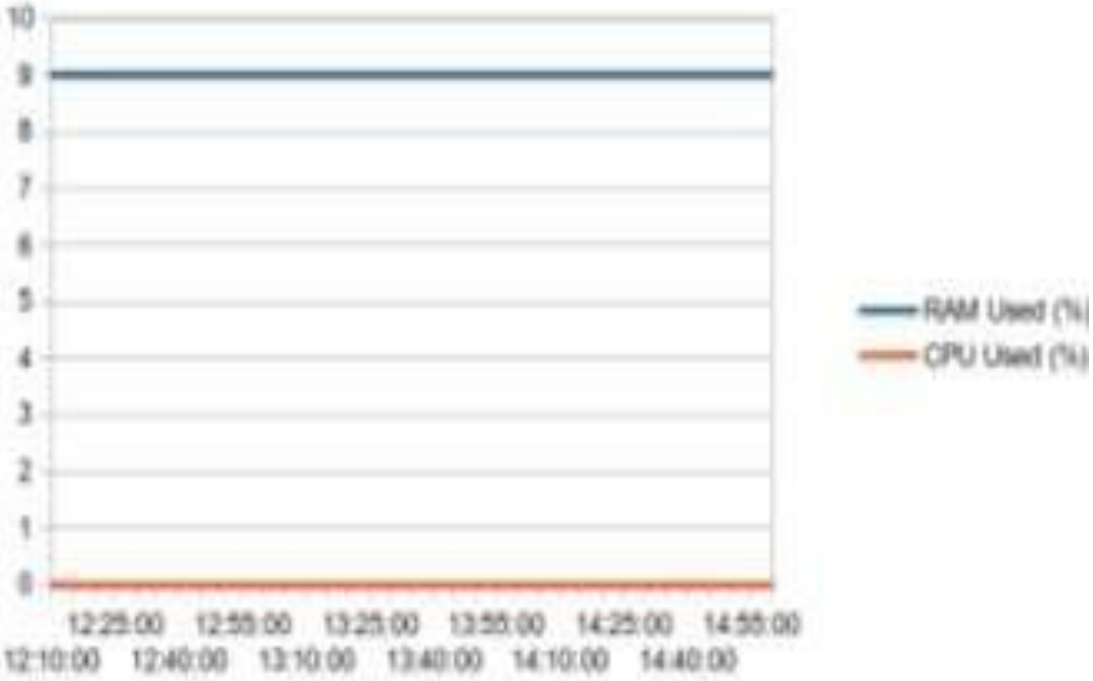
Dari tabel diatas terdapat besaran *error* antara pembacaan secara realtime dengan hasil pemantauan. Faktor utama yang menyebabkan perbedaan hasil pemantauan antara aplikasi *df*, *bmon* dan *top* dengan aplikasi *Network monitoring tools* memiliki *delay* waktu pengambilan. Ketiga aplikasi tersebut memberikan informasi secara *real time,* sedangkan aplikasi *Network Monitoring Tools* mengambil data dengan kecepatan berubah-ubah dengan nilai maksimum 20 detik sekali. Hal ini dikarenakan MIB pada SNMP *daemon* hanya mengupdate *database*nya dengan rate berkisar 15-20 detik. Apabila data diambil kurang dari rate tersebut maka data yang didapatkan akan selalu sama. Faktor lainnya yang mempengaruhi perbedaan hasil pemantauan adalah sistem pembulatan besaran yang didapat.

4.2. Pengujian perbandingan hasil pemantauan dengan aplikasi Cacti

Pada pengujian ini hasil pemantauan aplikasi WLTI *Network Monitoring Tools* akan dibandingkan dengan hasil pemantauan aplikasi Cacti. Pengujian ini bertujuan untuk mengetahui perbandingan kinerja aplikasi WLTI *Network Monitoring Tools* dengan aplikasi yang telah digunakan secara umum. Berikut perbandingan data hasil pemantauan *server* yang diambil secara bersamaan dengan kondisi *server idle* :



Gambar 2. Grafik data hasil pemantauan RAM dan CPU server oleh cacti



Gambar 3. Grafik data hasil pemantauan RAM dan CPU server oleh WLTI NMT

Dari kedua perbandingan diatas keanehan muncul pada data hasil pemantauan cacti. Secara logis kondisi *server* yang *idle* memiliki besaran penggunaan RAM dan CPU berada pada *level* terendah dan stabil (tidak berubah-ubah). Sedangkan pada grafik hasil pemantauan cacti (gambar 2) dapat dilihat terdapat fluktuasi penggunaan *resource server* yang terus meningkat. Sedangkan pada aplikasi WLTI penggunaan RAM dan CPU stabil di*level* 9% dan 0%. Hal ini logis mengingat kondisi *server* yang *idle* dan dapat dicocokkan dengan aplikasi *monitoring* bawaan linux seperti *top*. Diluar dari kompleksitas, fitur-fitur dan cakupan entitas jaringan yang mampu dipantau oleh cacti, dari kedua grafik perbandingan diatas dapat disimpulkan bahwa aplikasi WLTI NMT memiliki akurasi pemantauan yang lebih baik dari cacti. Selain itu aplikasi

WLTI NMT mampu melakukan pengambilan data dengan interval hingga 20 detik sedangkan cacti hanya menyediakan interval di 5 menit.

5. Kesimpulan

Dari penelitian yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut ; Persentase *error* pembacaan data oleh aplikasi sebesar 3.74% untuk CPU *Usage,* 11.57% untuk RAM *Usage*, 17.64% untuk *Traffic* IN dan 10.63% untuk *Traffic* Out. Hal ini dipengaruhi oleh *interval* pengambilan data dan sistem pembulatan besaran data. Sedangkan persentase *error* pembacaan data untuk *disk usage* dan *disk free* sebesar 0%. Hasil pengujian perbandingan antara WLTI NMT dengan Cacti menunjukkan bahwa pada kondisi *server* istirahat *(idle)* data hasil pembacaan oleh Cacti menampilkan grafik penggunaan RAM yang cenderung naik dan grafik penggunaan CPU yang konstan. Sedangkan data hasil pembacaan oleh WLTI NMT menampilkan grafik penggunaan RAM dan CPU yang konstan. Hal ini berarti aplikasi WLTI *Network Monitoring Tools* lebih akurat dibanding Cacti. Pengambilan data oleh aplikasi *Network Monitoring Tools* yang mampu bekerja dengan interval maksimum sebesar 20 detik sekali, sedangkan Cacti hanya mampu bekerja dengan interval sebesar 5 menit sekali. Hal ini berarti bahwa WLTI *Network Monitoring Tools* mampu menyajikan data lebih akurat. *Agent* JAVA berbasis JADE dapat diimplementasikan sebagai pemantau *server* dengan memanfaatkan *library SNMP4j* sebagai penyedia layanan akses ke protokol SNMP.

Daftar Pustaka

[1] H. Sajati, “Memonitor Server Dengan Cacti,” *Academia.edu*.
[2] “Pengawasan Jaringan Berbasis Web.” 2007. [Online]. Available: ftp://ftp.gunadarma.ac.id/linux/magazine/infolinux/2007/InfoLINUX_07-2007/38-41_Alternatif_07.pdf. [Accessed: 03-May-2016].
[3] A. M. Shiddiqi and A. P. Nugraha, “Sistem Monitoring Jaringan Dengan Protokol Snmp.” 2011.
[4] Cisco, “Simple Network Management Protocol.” pp. 1–8, 2013.
[5] D. B. Lange and M. Oshima, “Programming and deploying Java mobile agents with Aglets,” in *IBM Japan*, 1998, p. 225.
[6] F. Bellifemine, G. Caire, and D. Greenwood, *Developing multi agent systems with jade*. London: John Wiley & Sons. Ltd, 2004.