JEITECH – Journal of Electrical Engineering and Information technology
Department of Electrical Engineering University of Mataram
Vol. 2, No. 3, November 2024, pp. 20-28 , E-ISSN: 2987-7830

Sinkronisasi Data *Server-Client* dengan *Network File System* Berbasis *BashShell* melalui Jaringan *Virtual Private Network*

Mardiana Rizka Harsani[1], Giri Wahyu Wiriasto[2] , L.A.Syamsul Irfan A[3]
[1,2,3]Teknik Elektro Universitas Mataram, Jalan Majapahit no.62 Kota Mataram (83125), NTB - Indonesia

ARTICLE INFO

Article history (8 pt):

Received: November 19, 2024
Revised : November 28, 2024
Accepted : November 28, 2024

Keywords (8 pt):

Sinkronisasi data
Skrip BashShell
Network File System
Virtual Frivate Network
Obs. Geomagnetik Lombok

ABSTRACT (10 PT)

Virtual Private Network merupakan jaringan privat yang memanfaatkan jaringan publik untuk berbagi dan mengakses informasi dari lokasi mana pun, sementara layanan *Network File System* digunakan untuk mempermudah akses data. Terdapat Observatorium Geomagnetik Lombok (OGL), sebagai pusat pencatatan data-log oleh server pengukuran medan magnet bumi, berjarak lokasi dengan kota Mataram 42 Km. Untuk memenuhi kebutuhan akses data real-time, diperlukan sinkronisasi dan duplikasi data pada server OGL ke komputer klien di kampus Unram. Penelitian ini membuat program sinkronisasi dan duplikasi data berbasis *BashShel*l untuk ujicoba pada prototipe jaringan dengan VPN, memungkinkan data tersinkronisasi secara efisien dan dapat diakses.

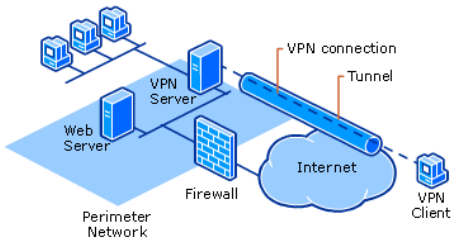
Corresponding Author:

Giri Wahyu Wiriasto, Teknik Elektro Universitas Mataram, Jalan Majapahit no.62 Kota Mataram, NTB - Indonesia
Email: giriwahyuwiriasto@unram.ac.id

1. PENDAHULUAN

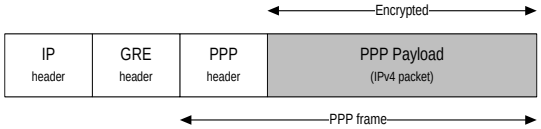
Membangun jaringan komputer [4][16] sangat krusial untuk mendukung kinerja Observatorium Geomagnetik Lombok (OGL) yang terletak di desa Rembitan, Kabupaten Lombok Tengah. Data yang dimiliki OGL perlu diolah oleh pihak luar seperti Universitas Mataram (UNRAM). Dengan jarak sekitar 42 km antara OGL dan UNRAM, jaringan komputer menjadi solusi efektif untuk memastikan komunikasi yang lancar antara kedua pihak. Namun, tantangan muncul dengan kondisi cuaca ekstrem dan biaya tinggi jika jaringan kabel dibangun antara OGL dan UNRAM. Selain itu, kebutuhan untuk memantau data real-time di server OGL mengharuskan pihak pengolah data dapat mengakses informasi tersebut dengan cepat. Oleh karena itu, diperlukan sinkronisasi data antara server OGL dan pihak pengolah di UNRAM, memungkinkan perubahan data dapat dipantau secara berkala. Untuk itu, dibutuhkan sistem yang memanfaatkan jaringan publik untuk memastikan komunikasi dan akses data antara OGL dan UNRAM dapat dilakukan dengan efisien.
VPN (*Virtual Private Network*) alternatif solusi dari permasalahan diatas.VPN (*Virtual Private Network*) [5] merupakan suatu bentuk jaringan *private*yang melaluijaringan *public* sehingga memungkinkan untuk berbagi dan mengakses informasi dari manapun.Berdasarkan paparan diatas, maka masalah yang ditemukan dapat dirumuskan sebagai berikut (i) Bagaimana membuat *tunneling* antar *client* agar *client* yang sebelumnya tidak bisa berkomunikasi menjadi dapat saling berkomunikasi?; (ii) Bagaimana membuat program sinkronisasi data yang ter-*schedule* secara otomatis?. Penelitian disini bertujuan untuk membuat program sinkronisasi data ter-*schedule* yang melalui sebuah *tunneling* yang dibuat pada jaringan komputer. Setelah penelitian ini terlaksana, diharapkan pada tahapan akhir didapatkan manfaat-manfaat penelitian sebagaimana berikut : Mendapatkan kesimpulan mengenai pengaplikasian VPN (*Virtual Private Network*) untuk sikronisasi data ter-*schedule*; Dapat digunakan sebagai *prototype* Observatorium Geomagnetik Lombok dalam melakukan *backup* dan sinkronisasi data.

VPN (*Virtual Private Network*)
Penjagaan keamanan data pada internet bervariasi, tergantung pada jenis aplikasi yang digunakan dan layer yang diamankan. Misalnya saja pengamanan pada layer *network*, berupa IP *security*, yang menggunakan proses autentifikasi *header* (*header authentication*) pada datagram IP dan proses enkripsi pada datagram. Pengamanan pada jaringan komputer dapat dilakukan dengan VPN (*Virtual Private Network*) [12] pada gambar 1, agar sejumlah komputer pada jaringan internet bisa membentuk jaringan *private* khusus [6][7][11].



Gambar 1. *Virtual Private Network*

***Point to Point Protocol* (PPTP)**
Protokol ini beroperasi pada *layer* 2 pada model OSI. Pada proses enkapsulasi, PPTP [13] mengenkapsulasi PPP *frames* pada IP *datagrams* untuk ditransmisikan pada jaringan. PPTP juga menggunakan koneksi TCP untuk mengelola *tunnel* dan GRE (*Generic RoutingEncapsulation*). Proses *tunneling* pada PPTP terjadi dengan cara membungkus paket informasi untuk kemudian ditransmisikan melalui jaringan internet seperti yang ditunjukkan pada Gambar 2. Pada proses ini PPTP menggunakan koneksi TCP yang dikenal sebagai PPTP *control connection* untuk menciptakan, merawat dan mengakhiri *tunnel* serta *Generic Routing Encapsulation* (GRE).



Gambar 2 Struktur PPTP Paket yang mengandung IPv4 Konten

Dalam hal enkripsi,pada gambar 2 tampak struktur PPTP menggunakan mekanisme otentikasi yang sama dengan PPP seperti *Extensible Authentication Protocol* (EAP), *Challange Handshake Protokol* (CHAP), *Shiva Password AuthenticationProtocol* (SPAP) dan *Password Authentication Protocol* (PAP) [5].

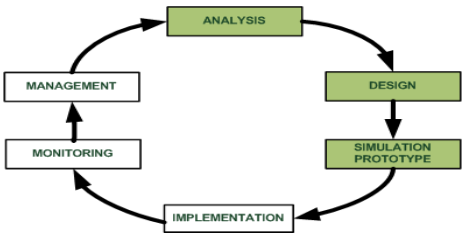
NFS
NFS memungkinkan untuk melakukan *share*, *export*/*import* data dari dan ke komputer kita melalui lingkungan jaringan. NFS hanya dapat digunakan dengan sistem operasi berbasis UNIX (GNU/Linux) [1][16].

Pemrograman *Bash*
Bash (*Bourne Again Shell*) merupakan 'turunan' dari *Bourne Shell* (sh), dan merupakan standar *shell* di Linux. *Bash* memiliki beberapa daya tarik, seperti pengeditan baris perintah, pelengkapan perintah (*commandcompletion*) dan pemanggilan ulang perintah-perintah yang pernah diketikkan. Selain itu *Bash* juga dapat dijadikan sebagai bahasa pemrograman yang terstruktur [10].

MD5
Algoritma MD5 digunakan untuk memeriksa integritas suatu data/*file*, seperti apakah suatu *file* yang diunduh dari *server* berada dalam keadaan utuh, atau apakah *file* yang disimpan di dalam *file* terkompresi tidak korup. MD5 *hash* juga dapat digunakan untuk menemukan adanya *file* duplikasi di dalam *harddisk*, dengan cara membandingkan nilai MD5 dari beberapa *file* itu sendiri. Jika *file –file* tersebut ternyata memiliki nilai MD5 yang sama, berarti *file* merupakan *file* terduplikasi. Contoh:
"*The quick brown fox jumps over the lazy dog*" = 9e107d9d372bb6826bd81d3542a419d6
"*The quick brown fox jumps over the lazy dog***.**" = e4d909c290d0fb1ca068ffaddf22cbd0

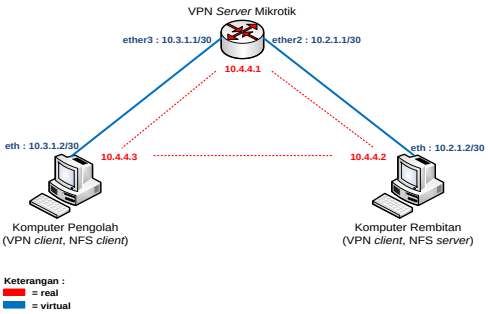
2. METODE

Dalam penulisan skripsi ini penulis menggunakan metode *Network Development Life Cycle* (NDLC). NDLC pada gambar 3 adalah metode yang digunakan untuk pengembangan dan membangun jaringan komputer. Ini merupakan kondisi model yang mendefinisikan siklus proses perancangan atau pengembangan suatu sistem jaringan komputer. NDLC terdiri dari 6 tahapan yaitu *analysis, design, simulation prototyping, implementation, monitoring* dan *management*.
Dari keenam tahapan yang terdapat pada metode NDLC diatas penulis hanya menggunakan 3 tahapan antara lain *analysis, design* dan *simulation prototype.*



Gambar 3 Metode dalam Penelitian

Perancangan suatu topologi jaringan [2][14] merupakan hal utama yang dibutuhkan seorang *network_administrator* dalam membangun sebuah simulasi jaringan.Simulasi jaringan dibangun untuk melihat kebutuhan suatu jaringan, dalam hal ini *network_administrator* dapat mengkoordinasi dan *management_network* sebelum jaringan nyata dibangun. Simulasi pada penelitian ini ditunjukkan pada Gambar 4.



Gambar 4 Desain Simulasi

Pengalamatan IP *address* [3][8][9] merupakan suatu identitas penomoran suatu *interface*. Tabel 1 berikut merupakan pengalamatan IP masing-masing *interface* pada simulasi VPN.

Tabel 1 Pengalamatan IP *Address*

No.	Keterangan	IP *Address*	*Interface*
1.	Mikrotik ke Komputer Rembitan	10.2.1.1/30	Ether2
2.	Mikrotik ke Komputer Pengolah	10.3.1.1/30	Ether3
3.	Komputer Rembitan ke mikrotik	10.2.1.2/30	Ethernet
4.	Komputer Pengolah ke mikrotik	10.3.1.2/30	Ethernet

Disamping itu, pengalamatan IP *address* yang akan digunakan pada saat komunikasi *private* (*local* IP *address* melalui VPN) antara *client* dan *server* terjalin ditunjukkan pada Tabel 2.

Tabel 2 Pengalamatan IP *AddressLocal* VPN

No.	Keterangan	IP *Address*	*Interface*
1.	Mikrotik VPN server	10.4.4.1	ppp0
2.	Komputer Rembitan	10.4.4.2	ppp0
3.	Komputer Pengolah	10.4.4.3	ppp0

Berikut algoritma **pseudocode** yang dibuat untuk program sinkronisasi data :

- Memeriksa keberadaan folder

```
IF folder1 exist THEN
continue process
ELSE
MAKE folder1
ENDIF
```

Gambar 5. Pseudocode pengecekan folder

- Membuat *listfile*

```
INTO folder1
MAKE list_file_folder1 THEN ascending

INTO folder2
MAKE list_file_folder2 THEN ascending

COMBINE list_file_folder1 AND list_file_folder2 TO list_all
```

Gambar 6. Pseudocode membuat file

- *Copy new file*

```
INTO folder1

FIND unique_line from list_all
MAKE unique_line TO list_new_file

WHILE read list_new_file
COPY new_file TO folder2
ENDWHILE
```

Gambar 7. Pseudocode meng-copy file

- *Differ file*

```
WHILE read file FROM list_file_folder1
m1=md5_file_folder1
m2=md5_file_folder2

IF m1≠m2 THEN
        COPY file TO folder2
ENDIF
ENDWHILE
```

Gambar 8. Pseudocode membandingkan isi data file

- *Move file*

```
no=1
WHILE read loop
no=no+1
INTO folder1
        MAKE list_file_folder1_no THEN ascending

INTO folder2
        MAKE list_file_folder2_no THEN ascending

COMBINE list_file_folder1_no AND list_file_folder2_no TO list_all_hps
```

```
FIND unique_line FROM list_all_hps
        MAKE unique_line TO list_hps

WHILE read list_hps
        MOVE move_file TO backup_folder
ENDWHILE
ENDWHILE
```

Gambar 9. Pseudocode memindahkan file

3. HASIL DAN DISKUSI

Pada sistem yang dibangun, proses pertama yang dilakukan adalah melakukan *tunneling* pada saat *booting.*Dan pada program sinkronisasi yang dibangun, proses pertama yang dilakukan adalah melakukan pengecekan folder-folder yang akan digunakan selama proses sinkronisasi seperti folder sumber, folder tujuan, folder *backup*, dan juga folder listdir. Apabila ternyata folder tersebut tidak ada pada saat pengecekan, maka program akan membuat folder tersebut. Sebaliknya, apabila pada saat pengecekan folder-folder tersebut telah ada, maka proses akan dilanjutkan. Gambar 10 menunjukkan sintaks yang digunakan untuk melakukan pengecekan folder-folder tersebut.

```
if [ ! -d $dir1 ]; then
echo "Folder mount sumber tidak terdeteksi !"
mkdir -p $dir1
chmod 777 $dir1
else
echo "Folder sumber $dir1 terdeteksi - proses dilanjutkan ... \n"
if test "$(ls "$dir1" 2>/dev/null)"; then
        echo "File yang akan disinkronisasi terdeteksi - proses dilanjutkan ... \n"
else
        echo "File yang akan disinkronisasi tidak terdeteksi !"
        echo "Keluar ..."
        exit
fi
fi
```

Gambar 10 Sintaks Pengecekan Folder

Untuk folder listdir, penamaan folder akan mengikuti tanggal tiap program sinkronisasi dieksekusi. Untuk folder sumber, apabila ternyata ditemukan bahwa pada folder sumber tidak berisikan *file* apapun (gagal melakukan *mount*), maka proses sinkronisasi dihentikan saat itu juga, tetapi jika folder sumber berisikan *file-file* yang telah di *mount* maka proses sinkronisasi akan dilanjutkan.

Pada folder listdir yang memuat *listfile* akan dibuat subfolder berdasarkan waktu (jam dan menit) tiap program sinkronisasi dieksekusi. Gambar 11 menunjukkan sintaks yang digunakan untuk membuat subfolder tersebut.

```
while [ -d $listdir/$jam-$no ]; do
no=$((no+1))
done
mkdir $listdir/$jam-$no
chmod 777 $listdir/$jam-$no
echo "folder list : $listdir/$jam-$no " >> $flog
```

Gambar 11. Sintaks membuat sub-folder

Selama proses sinkronisasi, untuk melakukan *copyfile*, *movefile* maupun mengecek *differfile* antara folder sumber dan folder tujuan menggunakan *listfile* yang dibuat dari kedua folder tersebut. Gambar 12 menunjukkan sintaks yang digunakan untuk membuat *listfile* dari folder sumber dan folder tujuan :

```
cd $dir1
find . -name '*' | cut -n -b 3- | sort -f > $listdir/$jam-$no/list_s.txt
```

```
cd $dir2
find . -name '*' | cut -n -b 3- | sort -f > $listdir/$jam-$no/list_t.txt

cat $listdir/$jam-$no/list_s.txt $listdir/$jam-$no/list_t.txt >> $listdir/$jam-$no/list_all.txt
```

Gambar 12 Sintaks *membuat file*

Setelah membuat *listfile* dari sumber dan folder tujuan dilanjutkan dengan melakukan *copynewfile* apabila terdapat *file* baru yang ter-*create* di komputer rembitan.*Newfile* akan di-*copy* jika ternyata ditemukan ketidakcocokan antara *listfile* folder sumber dan *listfile* folder tujuan pada saat mencocokkan *listfile* dari kedua folder tersebut (terdapat *listfile* lebih pada *listfile* folder sumber). Gambar 13 menunjukkan sintaks yang digunakan untuk melakukan pengecekan *newfile* sekaligus *copynewfile* :

```
sort -f $listdir/$jam-$no/list_all.txt | uniq -u > $listdir/$jam-$no/filebaru.txt

cd $dir1
while read line; do
if [ ! -e $dir2/$line ] && [ -e $line ]; then
        if test "$(find $line -maxdepth 0 -type d -empty 2>/dev/null)"; then
                echo "$line merupakan folder kosong yang akan dibuat di $dir2" >> $flog
                mkdir -p $dir2/$line

                elif [ -f $line ]; then
                        cp --parents $line $dir2 2>/dev/null
                        echo "File $line (`md5sum $line | awk '{print $1}'`) telah dicopy" >> $flog

                elif [ -d $line ] && test "$(ls -A "$line" 2>/dev/null)"; then
                        echo "" >> $femp
        fi
fi
done < $listdir/$jam-$no/filebaru.txt
```

Gambar 13 Sintaks *Copy New File*

Proses selanjutnya yakni melakukan pengecekan *differfile*. Pada proses ini pengecekan dilakukan dengan membandingkan MD5 *hash* yang dimiliki oleh tiap *file*. Dengan nama*file* yang sama pada folder sumber dan folder tujuan, dibandingkan MD5 *hash* yang dimiliki oleh *file* dari kedua folder tersebut. Apabila MD5 *hash* yang dihasilkan berbeda antara *file* di folder sumber dan *file* di folder tujuan, maka *file* tersebut akan di-*copy* ulang (*overwrite*). Gambar 14 menunjukkan sintaks yang digunakan untuk mengecek *differfile*.

```
cd $dir1
while read line; do
if [ ! -z $line ]; then
        if test "$(find $line -maxdepth 0 -type d -empty 2>/dev/null)"; then
                echo "" >> $femp

                elif [ -f $line ]; then
                        m1=$(md5sum $line | awk '{print $1}')
                        m2=$(md5sum $dir2/$line | awk '{print $1}')

                        if [ "$m1" != "$m2" ]; then
                                cp --parents $line $dir2 2>/dev/null
                                echo "File $line ($m1) dengan $m2 dicopy karena filenya berbeda"
>> $flog
                        fi

                elif [ -d $line ] && test "$(ls -A "$line" 2>/dev/null)"; then
                        echo "" >> $femp
        fi
fi
done < $listdir/$jam-$no/list_s.txt
```

Gambar 14 Sintaks *Differ File*

Setelah melakukan pengecekan *differfile*, selanjutnya dilakukan proses *movefile*. Pada proses ini, suatu *file*akan di-*move* dari folder tujuan ke folder *backup* apabila ternyata ditemukan adanya *file* berlebih di folder tujuan yang tidak ada di folder sumber. *File* berlebih tersebut ditemukan dengan membandingkan *listfile* yang terbentuk dari folder sumber dan folder tujuan. Gambar 15 menunjukkansintaks yang digunakan untuk melakukan proses *movefile*.

```
while read line;do

while [ -f $listdir/$jam-$no/list_all_hps_$nos.txt ]; do
        nos=$((nos+1))
done

cd $dir1
find . -name '*' | cut -n -b 3- | sort -f > $listdir/$jam-$no/list_s_hps_$nos.txt

cd $dir2
find . -name '*' | cut -n -b 3- | sort -f > $listdir/$jam-$no/list_t_hps_$nos.txt

cat $listdir/$jam-$no/list_s_hps_$nos.txt $listdir/$jam-$no/list_t_hps_$nos.txt >> $listdir/$jam-
$no/list_all_hps_$nos.txt

sort -f $listdir/$jam-$no/list_all_hps_$nos.txt | uniq -u > $listdir/$jam-$no/list_hps_$nos.txt

cd $dir2
while read line; do
        if test "$(find $line -maxdepth 0 -type d -empty 2>/dev/null)"; then
                cd $dir2
                cp --parents $line $back 2>/dev/null
                echo "Direktori kosong dihapus $dir2/$line" >> $flog
                rmdir $line

                elif [ -f $line ]; then
                        cd $dir2
                        cp --parents $line $back 2>/dev/null
                        echo "File $dir2/$line (`md5sum $line | awk '{print $1}'`) dipindahkan ke
folder backup" >> $flog
                        rm $line
        fi
done < $listdir/$jam-$no/list_hps_$nos.txt

done < /home/pengolah/sinc/loop.txt
```

Gambar 15 Sintaks *Move File*

Proses selanjutnya yakni membuat *listfile* terbaru dari folder sumber dan folder tujuan setelah melakukan 3 proses utama untuk melakukan sinkronisasi baik itu *copynewfile*, *differfile* maupun *movefile* untuk memastikan apakah *file-file* yang berada di folder tujuan telah sinkron dengan *file-file* di folder sumber. Hasil dari mencocokkan *listfile* kali ini (yang disimpan pada *file*cek.txt) nantinya akan digunakan untuk melaporkan status sinkronisasi apakah sinkronisasi yang dilakukan berhasil atau tidak. Berikut pada gambar 16 merupakan sintaks yang digunakan untuk membuat *listfile* sekaligus melaporkan status keberhasilan sinkronisasi :

```
cd $dir2
find . -name '*' | cut -n -b 3- > $listdir/$jam-$no/list_t_stlh.txt

cat $listdir/$jam-$no/list_s.txt $listdir/$jam-$no/list_t_stlh.txt >> $listdir/$jam-$no/list_all_stlh.txt

sort $listdir/$jam-$no/list_all_stlh.txt | uniq -u > $listdir/$jam-$no/cek.txt

if [ -s $listdir/$jam-$no/cek.txt ]; then
while read line; do
```

```
            if [ -f $dir2/$line ]; then
                    echo $line >> $listdir/$jam-$no/cek_t.txt
            fi
done < $listdir/$jam-$no/cek.txt
fi

# MEMBERITAHU USER APAKAH BERHASIL ATAU TIDAK
if [ -s $listdir/$jam-$no/cek_t.txt ]; then
echo "Ada file berlebih di $dir2" >>$flog
elif [ ! -s $listdir/$jam-$no/cek_t.txt ]; then
echo "...Sinkronisasi berhasil..." >>$flog
fi
echo "Akhir sync : `date`" >> $flog

exit
```

Gambar 16 Sintaks Status Akhir Proses

Status sinkronisasi berhasilakan muncul apabila hasil pengecekan *listfile* terakhir sama. Hal ini ditunjukkan dengan kosongnya isi *file* pada *file*cek_t.txtyang merupakan hasil dari pencocokan *listfile* terakhir dari folder sumber dan folder tujuan. Tetapi apabila pada hasil pengecekan *listfile* terakhir uditemukan adanya perbedaan maka sinkronisasi tidak berhasil dilakukan dan nama*file* berlebih tersebut dapat dilihat pada filecek_t.txt.

4. KESIMPULAN

Berdasarkan penelitian yang dilakukan, maka dapat diambil beberapa kesimpulan sebagai berikut : VPN (*Virtual Private Network*) dapat digunakan untuk mengatasi masalah koneksi jaringan lokal yang jauh sehingga antar *client* menjadi bisa saling berkomunikasi yang awalnya tidak bisa saling berkomunikasi satu sama lainnya; Program yang dibangun untuk sinkronisasi data berjalan dengan cukup baik secara terjadwal. Hal ini dapat dilihat pada kemampuan program dalam mensinkronisasi data pada komputer pengolah dan komputer rembitan dengan membandingkan parameter *size*, *datemodified*, dan MD5 *hash* masing-masing *file*; Berdasarkan data yang di dapat dari hasil pengujian, hasil uji *new file* memerlukan waktu yang lebih sedikit (31 detik) bila dibandingkan dengan *move file* (32 detik) dan *differ file* (33 detik); Semakin banyak *new file* yang di *copy* maka semakin banyak pula waktu yang di butuhkan. Hal ini terlihat pada saat pertama kali program sinkronisasi di jalankan membutuhkan waktu 54 detik untuk >50 *file* sedangkan pada saat ujicoba ke-3 membutuhkan waktu 31 detik untuk 4 *file*.

Acknowledgments

Ucapan terimakasih disampaikan kepada pengelola Observatorium Geomagnetik Lombok Universitas Mataram atas penelitian bersama yang telah dilakukan.

5. DAFTAR PUSTAKA

[1] Anonim, ”Panduan Aplikatif Migrasi dari Windows ke Linux”. Yogyakarta: ANDI, 2004.

[2] Anonim, "Planning and Cabling Networks," diakses pada 15 Oktober 2014, [Online]. Tersedia: http://www.highteck.net/EN/Cabling/Planning_and_Cabling_Networks.html

[3] M. L. Baihaqi, "TCP/IP (Transmission Control Protocol/Internet Protocol)," diakses pada 16 Oktober 2014, [Online]. Tersedia: http://ilmukomputer.org/wp-content/uploads/2013/03/TCPIP-Transmission-Control-Protocol-Internet-Protocol.pdf

[4] R. Budi, “Introduction to Computer Networking”. Yogyakarta: Skripta Media Creative, 2011.

[5] J. Davies, "Chapter 14 - Virtual Private Networking," diakses pada 15 Oktober 2014, [Online]. Tersedia: http://technet.microsoft.com/en-US/enus/library/bb727019.aspx

[6] P. Dwiyanto, S. R. Rahmana, dan A. Rahman, "Studi Kasus Perancangan Intrusion Prevention System Terhadap Serangan Man-In-The-Middle pada Jaringan Lokal," Jurnal Universitas Bina Nusantara, 2009. [Online]. Tersedia: https://core.ac.uk/display/11510822

[7] A. S. Rachman, “Public Key Infrastructure: Sistem Keamanan Informasi”. Yogyakarta: Universitas Mataram Press, 2004.

[8] W. S. Raharjo, “Practical TCP/IP: Mendesain, Menggunakan, dan Troubleshooting Jaringan TCP/IP di Linux dan Windows (Jilid 2)”. Yogyakarta: ANDI, 2004.

[9] E. Sutanta, ”Komunikasi Data dan Jaringan Komputer”. Yogyakarta: Graha Ilmu, 2005.

[10] A. SyahPutra, ”Pemrograman Berbasis Linux”. Yogyakarta: ANDI, 2002.

[11] Thor, “Hacker's Biggest Secret: Zero-Knowledge Password”. Jakarta: Elex Media Komputindo, 2008.

[12] G. Pelealu, "Penerapan VPN sebagai Solusi Keamanan Komunikasi Data di Internet," Jurnal STMIK Bumigora Mataram, 2001.

[13] M. T. Roseno, "Analisis Perbandingan Protokol Virtual Private Network (VPN) – PPTP, L2TP, IPSEC," Jurnal Politeknik Negeri Sriwijaya Palembang, 2013.

[14] I. W. Sulastra, "Konfigurasi FreeNAS sebagai server Network Attached Storage (NAS) Untuk Manajemen Penyimpanan Data secara Terpusat," Jurnal STMIK Bumigora Mataram, 2012.

[15] I. Sofana, ”Membangun Jaringan Komputer”. Bandung: Informatika Bandung, 2008.

[16] I. Winarno, "Network File System (NFS) + Samba Server," *Jurnal Teknik Institut Teknologi Sepuluh Nopember, 2006.