Jurnal Indonesia : Manajemen Informatika dan Komunikasi
www.journal.amikindonesia.ac.id/jimik/ Vol 5 No 1, Januari (2024)
E-ISSN: 2723-7079, P-ISSN: 2776-8074

Pengujian *High Availability* pada *Asynchronous* DNS Berbasis *Restknot* menggunakan Algoritma *Round Robin*

Putu Agus Eka Pratama [1*], Putu Veda Andreyana [2], Putu Ramaditya Nurjana [3]

[1*,2,3] Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana, Kabupaten Badung, Provinsi Bali, Indonesia.

Email: eka.pratama@unud.ac.id [1*], putuveda@unud.ac.id [2], ramadityanurjana@gmail.com [3]

Histori Artikel:
Dikirim 3 Desember 2023; *Diterima dalam bentuk revisi* 23 Desember 2023; *Diterima* 30 Desember 2023; *Diterbitkan* 10 Januari 2024.

Abstrak

Keberadaan Domain Name System (DNS) melalui peran dari DNS Record, berfungsi untuk menterjemahkan alamat domain ke dalam alamat IP. High availability menjadi kondisi mutlak agar layanan dari DNS Server tetap tersedia dan berjalan dengan baik. Untuk mewujudkannya, diperlukan adanya asynchronous API pada DNS server dan manajemen data di DNS Record. Penelitian ini mendesain dan mengimplementasikan RESTKnot sebagai asynchronous API pada DNS Server untuk manajemen DNS record, disertai dengan pengujian high availability menggunakan teknik Load Balancing dan algoritma Round Robin (RR). Hasil pengujian menunjukkan bahwa RESTKnot dapat membantu proses manajemen DNS record secara dinamis, di mana record DNS dapat dilakukan melalui RESTKnot API dan RESTKnot CLI, untuk selanjutnya dikirim ke agen RESTKnot dan dieksekusi oleh Knot DNS. Hasil pengujian juga menunjukkan bahwa RESTKnot mampu mewujudkan high availability melalui skenario pengujian pengaksesan sistem sebanyak sepuluh kali, di mana trafik padat pada web server pertama dapat segera dialihkan ke web server kedua untuk mewujudkan high availability.

Kata Kunci: DNS Record; High availability; Load Balancing; RESTKnot; Round Robin (RR).

Abstract

The existence of the Domain Name System (DNS) through the role of the DNS Record, functions to translate domain addresses into IP addresses. High availability is an absolute condition so that the service from the DNS Server is still available and running well. To make this happen, it is necessary to have an asynchronous API on the DNS server and data management on the DNS record. This research designs and implements RESTKnot as an asynchronous API on DNS Server for DNS record management, accompanied by high availability testing using Load Balancing techniques and the Round Robin (RR) algorithm. The test results show that RESTKnot can help the DNS record management process dynamically, where DNS records can be made through the RESTKnot API and RESTKnot CLI, to be sent to the RESTKnot agent and executed by Knot DNS. The test results also show that RESTKnot can realize high availability through a ten-time system access test scenario, where heavy traffic on the first web server can be immediately diverted to the second web server to achieve high availability.

Keyword: DNS Record; High availability; Load Balancing; RESTKnot; Round Robin (RR).

1. Pendahuluan

Di dalam jaringan komputer, *Domain Name System* (DNS) berperan penting di dalam menterjemahkan dan memetakan alamat domain ke dalam alamat *Internet Protocol* (IP) atau sebaliknya. Pengguna (manusia) lebih mudah untuk memahami pengalamatan domain dalam bentuk kata dan kalimat, sedangkan komputer (mesin) lebih mudah untuk memahami pengalamatan IP dalam bentuk angka dan binari. Dengan demikian, DNS berfungsi sebagai sebuah database terdistribusi, untuk menyimpan data-data *hostname* dan *IP Address* dari setiap komputer atau server pada jaringan, sehingga memudahkan proses pemetaan dan penterjemahan dari pengalamatan domain ke pengalamatan IP maupun sebaliknya [1]. Terkait dengan fungsinya tersebut, maka salah satu komponen penting di dalam DNS adalah DNS *Record*. DNS *Record* berperan untuk mengenali, mencatat, dan memetakan perubahan dan penterjemahan dari alamat domain berupa URL (*Uniform Resource Locator*) ke dalam alamat berbasis *Internet Protocol* (*IP Address*). DNS *Server* mampu menterjemahkan domain yang diakses oleh pengguna menjadi IP Address dengan adanya DNS *Record*. Berdasarkan hal ini, maka DNS *Record* perlu untuk dimanajemen dengan baik. Pada dasarnya, DNS *Record* terdiri atas enam bagian, yaitu:

1) NS *Record* untuk mengidentifikasi DNS server yang mengatur DNS Record
2) A *Record* untuk menghubungkan domain ke IP Address (IPv4)
3) AAAA *Record* yang dikhususkan untuk IPv6
4) *CName Record* untuk identifikasi sub domain
5) MX Record untuk koneksi ke e-mail dan *Mail Server*
6) *Certification Authority Authorization* (CAA) *Record* yang mengacu kepada sertifikasi domain [2].

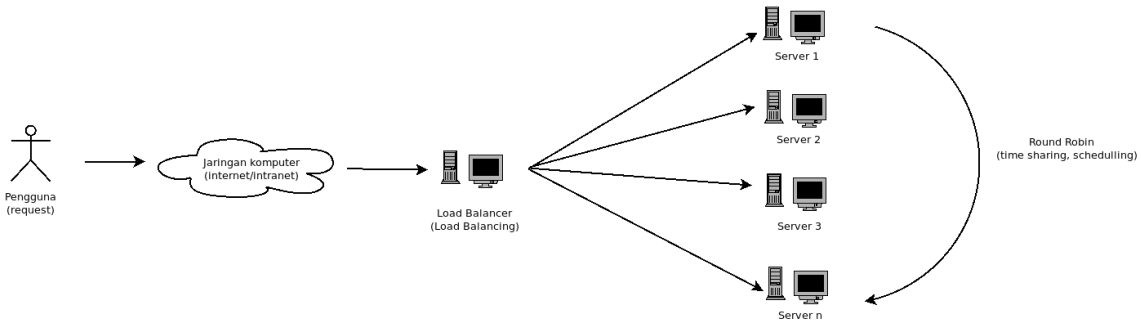
Keenam bagian ini memiliki peran penting terhadap fungsional dari DNS Server di dalam jaringan komputer. Dengan memperhatikan peran dari DNS *Server* dan DNS *Record*, maka sangat penting untuk memperhatikan kehandalan dari sebuah layanan pada DNS Server di dalam jaringan komputer serta manajemen data pada DNS *Record*. Terkait denga kehandalan pada jaringan komputer, dikenal adanya *availability* dan *high availability*. Pada jaringan komputer, *availability* merupakan sebuah kondisi di mana sebuah sistem mampu memastikan layanannya tetap tersedia, melalui adanya pemeriksaan dan penggunaan sumber daya komputasi sesuai dengan kebutuhan [3], sedangkan *high availability* merupakan upaya untuk mewujudkan *availability* pada sebuah sistem dengan meminimalisir adanya kegagalan penyediaan layanan, yang dilakukan dengan cara penyediaan duplikasi layanan atau sistem secara otomatis sebagai pengganti komponen saat terjadi masalah [4]. Dengan demikian, kondisi *availability* melalui *high availability*, perlu untuk diwujudkan pada sistem yang menyediakan layanan di jaringan komputer, termasuk juga dalam hal ini pada DNS *server*. Teknis yang umum digunakan untuk mewujudkan *availability* adalah *Load Balancing*. *Load Balancing* merupakan metode pada jaringan komputer yang berfungsi untuk mendistribusikan beban kerja pada dua atau lebih komputer atau server, sehingga layanan dapat berjalan optimal serta tidak mengalami kelebihan beban (*overload*) [5]. Namun *Load Balancing* memerlukan adanya infrastruktur yang besar, biaya yang lebih mahal, dan konfigurasi yang lebih rumit [6]. Diperlukan adanya kombinasi *Load Balancing* dengan *tool*, metode, dan algoritma lainnya.

Penelitian ini mendesain dan mengimplementasikan kombinasi metode *Load Balancing* beserta dengan *asynchronous* API pada DNS *Server* dan algoritma penjadwalan (*Schedulling*). *Asynchronous* API pada DNS merupakan API yang khusus digunakan pada DNS *Server* untuk membantu menjalankan langsung tugas (*job*) selanjutnya (misal: request dari pengguna) tanpa perlu menunggu *job* sebelumnya tersebut selesai [7]. Dari definisi ini, maka sebuah *Asynchronous* API pada DNS dapat menjadi solusi untuk mewujudkan *availability* [7]. Pada penelitian ini, *RESTKnot* dipilih sebagai *asynchronous* API pada DNS *Server* untuk manajemen DNS *record*, di mana algoritma *Round Robin* (RR) dipilih sebagai algoritma penjadwalan pada metode *Load Balancing* yang digunakan.

Round Robin (RR) dipilih sebagai algoritma penjadwalan (*schedulling*)yang digunakan pada penelitian ini, karena memiliki kelebihan dari sisi kesederhanaan proses dan kemudahan untuk diimplementasikan di dalam jaringan komputer untuk manajemen beban trafik jaringan dan *request* dari pengguna [8]. Konsep kerja dari algoritma *Round Robin* adalah dengan cara membagi beban yang

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

diterima oleh *server* dari *request* pengguna secara berurutan untuk kemudian dialihkan ke *server-server* lainnya, memanfaatkan *time sharing*, sehingga antrian dapat dialihkan secara bergiliran [9]. Dengan demikian, *Round Robin* dapat diimplementasikan dengan baik pada jaringan komputer di sejumlah studi kasus. Gambar 1 menampilkan bagan jaringan yang menunjukkan kombinasi antara *Load Balancing* dan *Round Robin*:



Gambar 1. Kombinasi *Load Balancing* dan *Round Robin*

RestKnot DNS menjadi pilihan *software open source* untuk digunakan di dalam penelitian ini, yang menawarkan kelebihan berupa kemudahan implementasi, performansi tinggi dalam jaringan, *availability*, manajemen DNS *Record*, dan *high-level asynchronous* API ke DNS melalui *Knot* DNS [10]. *RESTKnot* terdiri atas tiga bagian, yaitu: *RESTKnot agent*, *RESTKnot API*, dan *RESTKnot CLI*. Data pada DNS *Record* dicatat melalui web API pada *RESTKnot* API, untuk kemudian diterjemahkan melalui *RESTKnot CLI* dan diteruskan oleh *RESTKnot agent ke Knot DNS*. Tujuan yang ingin dicapai di dalam penelitian ini yaitu mewujudkan *high availability* dan manajemen DNS *Record* pada DNS *Server* berbasis *asynchronous* API menggunakan *RESTKnot* melalui implementasi dan pengujian yang dilakukan. Dua buah pertanyaan penelitian yang menjadi rumusan masalah pada penelitian ini, yaitu:

1) Bagaimana teknis untuk mengimplementasikan *RESTKnot* pada DNS untuk mewujudkan *high availability*
2) Bagaimana cara untuk melakukan pengujian dan pengukuran high availability berdasarkan kepada *RESTKnot* yang diimplementasikan tersebut. Hipotesa penelitian ini yaitu bahwa implementasi *RESTKnot* mampu mewujudkan *high availability* pada DNS *server*.

Terdapat tiga belas penelitian sebelumnya yang melatar belakangi penelitian ini dan menjadi state of the art. Penelitian pertama menguraikan tentang CMQ sebagai sebuah *asynchronous message queue* berbasis UDP untuk mewujudkan *orkestrasi*, *messages*, *events* dan proses di dalam jaringan *cloud computing*, dengan tujuan untuk mewujudkan sistem pengiriman pesan yang handal di dalam jaringan [11]. Penelitian kedua membahas tentang implementasi internet sehat dengan menggunakan RouterBoard Mikrotik dan DNS *Filtering* berbasis *Open* DNS, sehingga dapat mencegah pengguna yang mencoba mengakses *website* atau situs yang diblokir [12]. Penelitian ketiga mengimplementasikan solusi berupa *Network Simulator*-2 (NS-2) dan Geo DNS untuk permasalahan *overload* pada *single server non Content Delivery Network* (CDN)[13].

Penelitian keempat mengemukakan tentang implementasi protokol *high availability* berupa HRSP (*Hot Standby Routing Protocol*) dan *First Hop Routing Protocol* (FHRP) pada studi kasus perbankan, yang di dalamnya meliputi perencanaan topologi jaringan komputer, penerapan protokol FHRP, teknik konfigurasi, simulasi, dan perbandingan keduanya, di mana hasil pengujian menunjukkan bahwa protokol HRSP memiliki kelebihan dibandingkan protokol FHRP dari sisi kehandalan pada penyediaan layanan perbankan di jaringan computer [14]. Penelitian kelima menyebutkan tentang pengembangan sebuah metode untuk mendeteksi malicious traffic pada infrastruktur DNS di dalam jaringan komputer dengan menggunakan algoritma Decision Tree, dengan tingkat akurasi sebesar 96.36%[15]. Penelitian keenam menguraikan tentang implementasi *standalone server* dan *clustering server* berbasis *Real Application Clustering* (RAC) dan Algoritma *Round Robin* disertai dengan analisa kinerja performansinya dari sisi *Quality of Service* (QoS), yang menunjukkan hasil *clustering server* memiliki

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

kehandalan dan nilai QoS yang lebih baik dibandingkan *standalone server* [16]. Penelitian ketujuh *High Availability* pada sistem *cluster* menggunakan *system sharding database* berbasis noSQL dan mongoDB, di mana hasil penelitian berhasil mengimplementasikan server yang diset dengan *High Availability*, *failover*, dan *clustering* [17]. Penelitian kedelapan mengimplementasikan replikasi antara pusat data dengan sistem *Disaster Recovery Center* (DRC) berbasis Metode *Synchronous* di mana hasil pengujian menunjukkan bahwa Metode *Synchronous* berhasil menyalin dan mendistribusikan data dari satu *database* ke *database* yang lain, kemudian mensinkronisasikan antar database untuk menjaga konsistensi data [18]. Penelitian kesembilan menguraikan *availability* pada jaringan dan DNS di studi kasus data center dengan memanfaatkan *Double Deep Packet Instrusion* [19].

Penelitian kesepuluh menjelaskan tentang implementasi *Cluster-based Load Balancing* (CLB) *server* untuk mengevaluasi tingkat keefektifan layanan DNS pada jaringan, dengan hasil akhir berupa peningkatan pengaksesan data sebesar 196.38% dan pengurangan *response time* transaksi sebesar 66.04% dibandingkan dengan *single server* [20]. Penelitian kesebelas membahas tentang *high availability cluster server* menggunakan metode heartbeat untuk menguji *failover clustering* pada *private cloud*, dengan hasil pengujian menunjukkan nilai rata-rata *throughput* sebesar 5,254 Mbps [21]. Penelitian kedua belas menguraikan tentang implementasi *High Availability Server* menggunakan metode *failover clustering* dan *replication mirror*, untuk mengantisipasi kemungkinan terganggunya layanan di jaringan maupun kerusakan *hardware* [22]. Penelitian ketiga belas membahas tentang keamanan pada server DNS di dalam jaringan dari bentuk serangan *Man in the Middle* (MitM) dengan menggunakan pengkalkulasian nilai-nilai QoS pada jaringan terhadap anomali pada trafik jaringan [23].

Ketiga belas penelitian tersebut turut membantu peta jalan penelitian yang telah dilakukan terkait dengan DNS *server* pada jaringan komputer. Berdasarkan kepada penelitian-penelitian tersebut, terdapat sebuah gap penelitian yaitu belum adanya penelitian mengenai pengujian *high availability* pada *asynchronous* DNS, sehingga pada penelitian ini gap tersebut diambil untuk dikerjakan dengan menggunakan *tool RestKnot* dan algoritma *Round Robin* pada jaringan.

2. Metode Penelitian

2.1 Alat dan Bahan

Pada penelitian ini digunakan sejumlah *hardware* dan *software* pendukung. Untuk kebutuhan *software* digunakan sistem operasi *Linux Ubuntu*, *RESTKnot*, *Apache*, dan *HaProxy* yang semuanya berlisensi *open source*. Untuk kebutuhan hardware digunakan komputer dengan spesifikasi: Dell Inspiron 15-5568 Intel i7-6500U (4) 3.100GHz, RAM 16GB. Selain itu juga digunakan koneksi internet untuk *online repository*.

2.2 Metode Penelitian

Metode penelitian yang digunakan di dalam penelitian ini adalah *Experimental Method*, dengan langkah-langkah meliputi: penentuan topik dan judul penelitian, perumusan masalah, implementasi, pengujian, dan Analisa [24].

2.3 Metodologi Penelitian

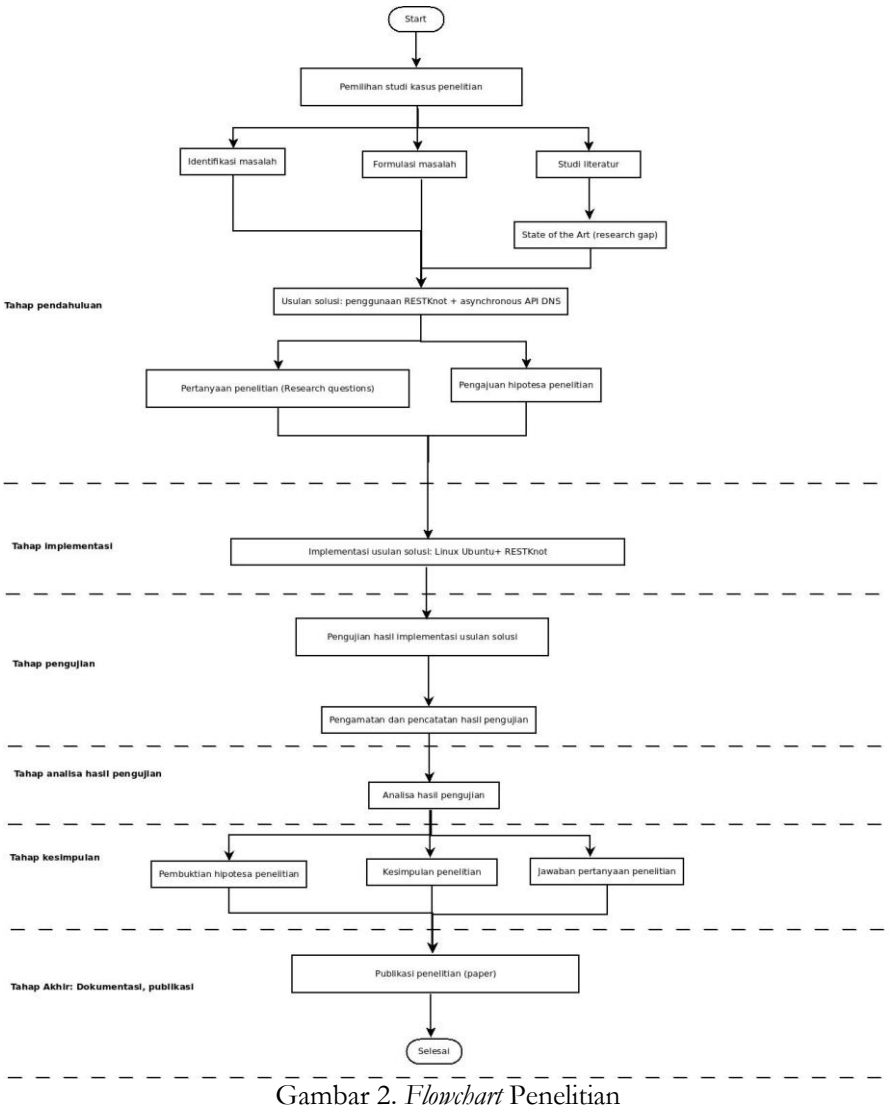
Metodologi penelitian yang digunakan di dalam penelitian ini adalah *Design Science Research Methodology* (DSRM) dengan delapan langkah terurut:

1) Latar belakang permasalahan, kajian teori, dan *state of the art*
2) Penentuan tujuan penelitian
3) Perumusan masalah
4) Usulan solusi dan desain
5) Implementasi
6) Pengujian
7) Dokumentasi dan analisa hasil pengujian
8) Publikasi hasil penelitian [25] [26].

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

2.4 *Flowchart* Penelitian

Di dalam penelitian ini, terdapat urutan tahapan langkah penelitian, yang disajikan ke dalam bentuk *flowchart* penelitian. Gambar 2. menunjukkan *flowchart* penelitian:



Gambar 2. *Flowchart* Penelitian

Berdasarkan kepada *flowchart* penelitian pada Gambar 2., terdapat enam tahapan di dalam penelitian ini, yang meliputi: tahap pendahuluan, tahap implementasi, tahap pengujian, tahap analisis hasil pengujian, tahap kesimpulan, dan tahap akhir (dokumentasi, publikasi). Pada tahap pendahuluan dilakukan terlebih dahulu pemilihan studi kasus penelitian, dalam hal ini terkait dengan *high availability* pada DNS server di jaringan komputer. Dilakukan identifikasi permasalahan, rumusan masalah dan formulasi masalah berupa pertanyaan-pertanyaan penelitian, serta studi literatur dari sejumlah paper publikasi sebagai *state of the art* dan *research gap*. Pada tahap implementasi, dilakukan proses instalasi dan konfigurasi sistem dari sisi sistem operasi *Linux Ubuntu*, *DNS Server*, *REST Knot*, dan *web server* yang disesuaikan dengan skenario pengujian yang dilakukan pasca tahapan implementasi, Pada tahapan pengujian, dilakukan pengujian di sisi pengembang (*Black Box Testing*) terhadap sistem DNS *Server* berbasis *RESTKnot*, untuk kemudian hasil pengujian diamati dan dicatat. Pada tahap analisa hasil pengujian, dilakukan analisa terhadap data-data hasil pengujian, dilanjutkan dengan pembuktian hipotesa penelitian, jawaban atas pertanyaan penelitian, kesimpulan, dan saran. Terakhir adalah tahapan publikasi penelitian dalam bentuk paper ke jurnal ilmiah.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

3. Hasil dan Pembahasan

3.1 Implementasi

Untuk implementasi dan pengujian, menggunakan tiga buah mesin dengan pengalamatan dan fungsi sebagai berikut:

1) DNS Server (IP Address 192.168.56.138, netmask 255.255.255.0, Gateway 192.168.56.255)
2) Web server pertama (IP Address 192.168.56.136, netmask 255.255.255.0, Gateway 192.168.56.255), 3.)Web server kedua (IP Address 192.168.56.137, netmask 255.255.255.0, Gateway 192.168.56.255).

Implementasi dimulai dengan melakukan instalasi dan konfigurasi sistem operasi *Linux Ubuntu*, *web server Apache*, *RESTKnot*, dan *HAProxy*. Pasca instalasi sistem operasi, repositori *Linux Ubuntu* dikonfigurasikan dengan menggunakan menggunakan perintah:

```
sudo nano /etc/apt/sources.list
```

Lalu dilanjutkan dengan perintah:

```
sudo apt-get update -y.
```

Selanjutnya untuk daemon web server Apache diaktifkan menggunakan perintah:

```
sudo /opt/lampp/lampp start
```

Untuk asynchronous API DNS berbasis RESTKnot, ditambahkan repositori online melalui PPA mengunakan perintah:

```
sudo add-apt-repository ppa:cz.nic-labs/knot-dns
```

Selanjutnya dilakukan instalasi RESTKnot menggunakan perintah:

```
sudo apt-get install knot -y --fix-missing
```

Pasca instalasi, service RESTKnot dijalankan menggunakan perintah:

```
sudo systemctl start knot
```

Status dari service RESTKnot dicek dengan menggunakan perintah:

```
sudo systemctl status knot
```

Load balancing dan *Round Robin* diimplementasikan dengan menggunakan *HAProxy*. Instalasi *HAProxy* dilakukan dengan menggunakan perintah:

```
sudo apt-get install haproxy -y -fix-missing
```

Perintah di atas akan menginstalasi paket HAProxy beserta dengan semua library yang diperlukan secara online dari repositori. Pasca instalasi, dilakukan sejumlah konfigurasi. Konfigurasi pertama dilakukan *HAProxy*, dengan *file* konfigurasi berada pada *path* lokasi */etc/haproxy/haproxy.cfg*. Konfigurasi dilakukan menggunakan perintah:

```
sudo nano /etc/haproxy/haproxy.cfg
File konfigurasi diberikan sejumlah nilai dan parameter berikut:
frontend web=http
bind *:80
mode http
default_backend apache
backend apache
mode http
balance round robin
server web1 192.168.56.136:80 check
server web2 192.168.56.137:80 check
```

Isian dan parameter konfigurasi di atas, menunjukkan bahwa web server 1 dengan IP Address

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

192.168.56.136 dan web server 2 dengan IP Address 192.168.56.137, keduanya berjalan pada port number 80 (web), telah sukses berjalan pada jaringan dengan frontend web pada protokol HTTP, backend Apache web server, dengan load balancing menggunakan algoritma Round Robin. Selanjutnya dilakukan konfigurasi pada *file knot.conf* dan *file zone* dari *RESTKnot* melalui RESTKnot CLI, yang bertujuan untuk mengatur zone Knot Server dan zone data saved, dengan menggunakan perintah:

```
sudo gedit /etc/knot/knot.conf
sudo gedit var/lib/knot/netcentricprinciples.com.zon
```

Pada konfigurasi *zone* ini, diinputkan IP Address untuk DNS *Server* (beserta *Load Balancing* dan HAProxy) di 192.168.56.138, subnet mask 255.255.255.0 dan IP Gateway di 192.168.56.255. Pasca konfigurasi *zone*, kemudian dilakukan pengecekan *zone* menggunakan *RESTKnot CLI* di Terminal menggunakan perintah *knotc zone-read* dan *knotc conf-read*. Gambar 3 menunjukkan konfigurasi *file* melalui *RESTKnot CLI*.

```
root@rama:/etc/knot# systemctl start knot
root@rama:/etc/knot# knotc reload
Reloaded
root@rama:/etc/knot# knotc conf-read
server.rundir = /run/knot
server.user = knot:knot
server.listen = 127.0.0.1@53 ::1@53
log.target = syslog
log[syslog].any = info
template.id = default
template[default].storage = /var/lib/knot
template[default].file = %s.zone
zone.domain = netcentricprinciples.com.
zone[netcentricprinciples.com.].file = netcentric.com.zone
root@rama:/etc/knot#
```

Gambar 3. *RESKnot CLI*

Untuk mengimplementasikan *high availability* pada DNS *Server*, dilakukan konfigurasi pada *file default.conf*, dengan menggunakan perintah:

```
sudo nano /etc/apache2/ites-availables/000-default.conf.
```

Perintah tersebut diberikan empat nilai dan parameter, yaitu:

1) ServerName netcentricprinciples.com
2) ServerAlias www.netcentricprinciples.com
3) ServerAdmin webmaster@localhost
4) DocumentRoot /var/www/html

Domain *netcentricprinciples.com* akan menjadi URL untuk alamat web server 1 dan web server 2 yang akan diakses pada saat pengujian *high availability*. Hasil yang diharapkan saat pengujian *high availability* adalah sistem mampu mengalihkan beban request akses pengguna dari web server 1 ke web server 2. Table 1 menampilkan ringkasan dari daftar isian untuk setiap file konfigurasi pada *Apache web server*, RESTKnot, dan *HAProxy*.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

Tabel 1. Daftar isian konfigurasi

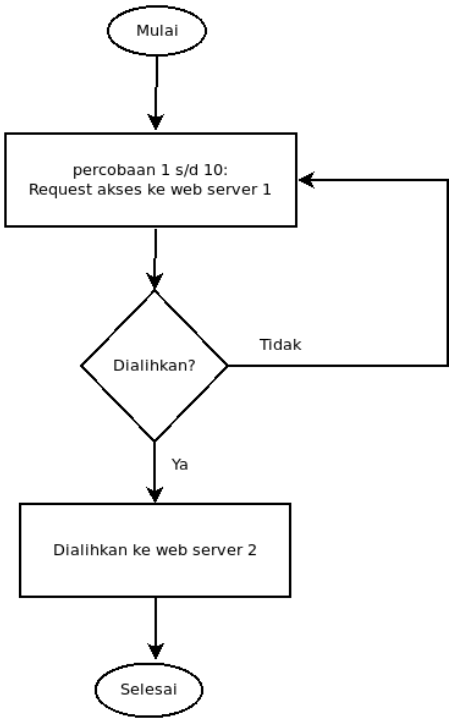
No	File	Isian Konfigurasi	Path
1	ServerName	netcentricprinciples.com	/etc/apache2/sites-availables/000-default.conf
2	ServerAlias	www.netcentricprinciples.com	/etc/apache2/sites-availables/000-default.conf
3	ServerAdmin	webmaster@localhost	/etc/apache2/sites-availables/000-default.conf
4	DocumentRoot	/var/www/html	/etc/apache2/sites-availables/000-default.conf
5	IP Address DNS Server (Load Balancing+HAProxy)	192.168.56.138	/var/lib/knot/netcentricprinciples.com.zone
6	IP Address Web Server 1	192.168.56.136	-
7	IP Address Web Server 2	192.168.56.137	-
8	HAProxy	frontend web=http bind *:80 mode http default_backend apache backend apache mode http balance round robin server web1 192.168.56.136:80 check server web2 192.168.56.137:80 check	/etc/haproxy/haproxy.cfg

Setiap isian konfigurasi pada Table 1 dilakukan pada *Apache web server* dan *RESTKnot* sesuai lokasi direktori/sub direktori (*path*) dari masing-masing *file* konfigurasi. Selain itu juga dilakukan pengalamatan jaringan untuk DNS *server* dan kedua *web server* yang saling terhubung.

3.2 Pengujian *High Availability*

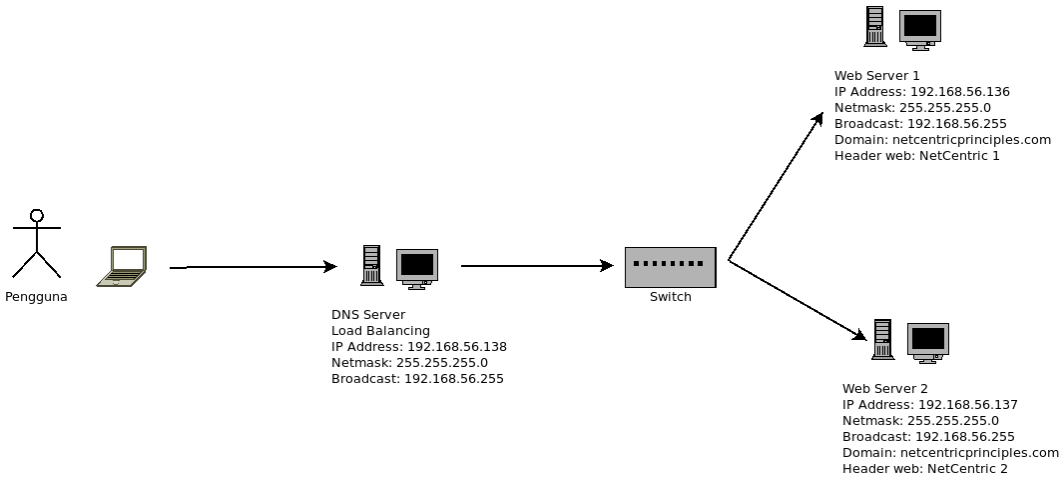
Metode Pengujian *high availability* pada penelitian ini dilakukan terhadap *web server* 1 untuk menguji beban *request* akses yang diterima dari pengguna, untuk dapat dialihkan ke web server 2. Pengujian dilakukan mengunakan metode *BlackBox Testing*, dengan sejumlah langkah-langkah skenario pengujian. Urutan langkah skenario pengujian disajikan dalam bentuk flowchart. Gambar 4 menunjukkan *flowchart* dari skenario pengujian.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582



Gambar 4. *Flowchart* dari Skenario Pengujian

Berdasarkan Gambar 4 *request* akses ke *web server* 1 dilakukan oleh pengguna sebanyak sepuluh kali. Apabila tidak terdapat pengalihan dari *web server* 1 ke *web server* 2, maka request akses ke *web server* 1 terus dilakukan. Apabila terjadi pengalihan dari *web server* 1 ke *web server* 2, maka pengujian berakhir. Untuk menunjang pengujian yang dilakukan, dikembangkan bagan pengujian di jaringan komputer, yang melibatkan *web server* 1, *web server* 2, dan DNS *server* dengan *REST Knot* dan *HAProxy* di dalamnya, untuk mengimplementasikan *Load Balancing* dan *Round Robin*. Gambar 5 menunjukkan bagan pengujian di jaringan komputer yang digunakan pada skenario pengujian.

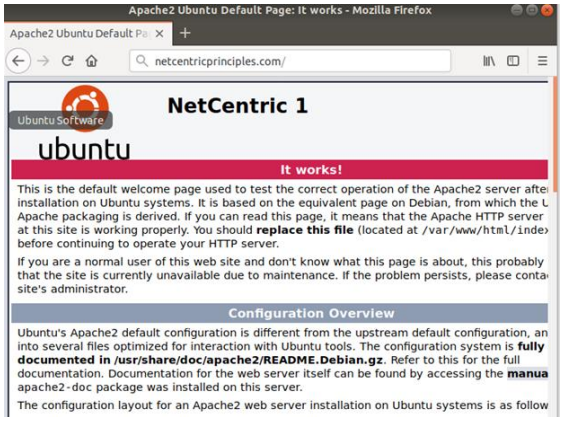


Gambar 5. Bagan Jaringan (*Load Balancing*, *Round Robin*)

Berdasarkan kepada Gambar 5 pengguna terhubung pertama kali ke *web server* 1 dan *web server* 2 melalui DNS *Server* dan *switch*. Selanjutnya, pengguna melakukan *request* akses ke *web server* 1 sebanyak 10 kali. Saat pengguna melakukan *request* akses ke *web server* 1 pada pengujian pertama, sistem menampilkan *home page web server* 1 kepada pengguna, dengan *header web tertera NetCentric* 1. Hal yang

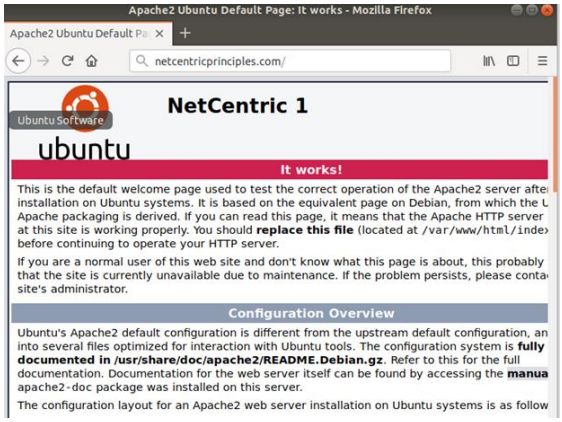
Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

sama juga terjadi pada pengujian kedua, ketiga, dan keempat. Gambar 6 menampilkan hasil pengujian pertama hingga keempat.



Gambar 6. Tampilan *Web Server* 1

Pada pengujian kelima, sistem mengalihkan pengguna dari *web server* 1 ke *web server* 2. Pada pengujian ini, ditampilkan *home page web server* 2 dengan *web header NetCentric* 2. Hal yang sama terjadi pada pengujian keenam, ketujuh, kedelapan, kesembilan, dan kesepuluh. Gambar 7 menampilkan hasil pengujian kelima hingga kesepuluh.



Gambar 7. Tampilan *Web Server* 2

3.3 Hasil Pengujian

Berdasarkan kepada pengujian yang telah dilakukan, pada skenario pengujian sebanyak sepuluh kali, diperoleh data-data hasil pengujian. Table 2 menunjukkan data-data hasil pengujian, yang meliputi aksi yang dilakukan, target, respon, dan status:

Tabel 2. Data Pengujian (Aksi, Target, Respon, Status)

No	Aksi	Target	Respon	Status
1	Request akses pertama	Web server 1	Pengguna dapat mengakses web server 1 dengan baik	Sukses
2	Request akses kedua	Web server 1	Pengguna dapat mengakses web server 1 dengan baik	Sukses
3	Request akses ketiga	Web server 1	Pengguna dapat mengakses web server 1 dengan baik	Sukses
4	Request akses keempat	Web server 1	Pengguna dapat mengakses web server 1 dengan baik	Sukses

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

5	Request akses kelima	Web server 1	Pengguna dialihkan dari web server 1 ke web server 2	Sukses
6	Request akses keenam	Web server 1	Pengguna dialihkan dari web server 1 ke web server 2	Sukses
7	Request akses ketujuh	Web server 1	Pengguna dialihkan dari web server 1 ke web server 2	Sukses
8	Request akses kedelapan	Web server 1	Pengguna dialihkan dari web server 1 ke web server 2	Sukses
9	Request akses kesembilan	Web server 1	Pengguna dialihkan dari web server 1 ke web server 2	Sukses
10	Request akses kesepuluh	Web server 1	Pengguna dialihkan dari web server 1 ke web server 2	Sukses

Berdasarkan kepada hasil pengujian di atas, terlihat bahwa sistem mampu melayani request akses dari pengguna ke *web server* 1 pada pengujian pertama hingga pengujian keempat. Kemudian sistem mampu mewujudkan *high availability* melalui *Load Balancing* dan *Round Robin* melalui pengalihan beban *request* akses dari pengguna ke *web server* 1 menuju *web server* 2 pada pengujian kelima hingga kesepuluh. Berdasarkan kepada pengujian yang dilakukan dan hasil-hasil pengujian, terlihat bahwa *RESTKnot* sebagai *assynchronous* API pada DNS server mampu mewujudkan *availability* melalui *high availability*. Hal ini disebabkan oleh karena metode *Load Balancing* beserta dengan algoritma *Round Robin* mampu mengalihkan beban *request* pengguna dari *web server* 1 ke *web server* 2 secara berurutan. Selain itu juga dilakukan pembagian waktu secara adil untuk distribusi beban (*time sharing*) berdasarkan kepada konsep dari *Round Robin*. Implementasi dari metode *Load Balancing* dan algoritma *Round Robin* dilakukan melalui *HAProxy* yang dijalankan beserta dengan *RESTKnot* untuk manajemen data DNS *record* (untuk *zone*). Konfigurasi yang dilakukan pada keduanya ini, membantu jalannya *Load Balancing* dan *Round Robin* pada pengujian yang dilakukan. Selain itu, konfigurasi pada *Apache web server* juga turut membantu di dalam penyediaan layanan pada kedua *web server* yang diujikan.

4. Kesimpulan

Berdasarkan kepada pengujian yang dilakukan, dapat diperoleh dua poin kesimpulan berikut:

1) Teknis implementasi *RESTKnot* pada DNS *Server* untuk mewujudkan *high availability* berhasil dilakukan dengan menambahkan *HAProxy* untuk mengimplementasikan *Load Balancing* dan *Round Robin* beserta dengan sejumlah konfigurasi.
2) Pengujian dan pengukuran *high availability* pada DNS *Server* berbaasis *RESTKnot* yang diimplementasikan tersebut, dilakukan melalui request akses ke dua buah *web server* yang telah dipersiapkan, di mana sistem mampu mengalihkan beban *request* akses dari pengguna ke *web server* 1 pada pengujian kelima hingga kesepuluh.

Hasil pengujian ini menunjukkan bahwa *RESTKnot* mampu mewujudkan *high availability* melalui skenario pengujian yang dijalankan, di mana sistem meminimalisir adanya kegagalan penyediaan layanan saat adanya request akses dari pengguna ke *web server* 1, melalui pengalihan secara otomatis ke duplikasi layanan atau sistem lainnya (*web server* 2). Berdasarkan hal ini, maka dapat dibuktikan bahwa *availability* melalui *high availability*, dapat diwujudkan dengan cara memanfaatkan pemanfaatan *asynchronous* API pada DNS *Server* dan manajemen DNS *Record* berbasis *RESTKnot*, disertai dengan kombinasi metode *Load Balancing* dan *Round Robin*. Ke depannya, penelitian ini dapat dilanjutkan dengan melakukan pengujian pada DNS Server dengan menggunakan metode dan algoritma lainnya pada jaringan *client server* dan *cloud*, sehingga diharapkan dapat memperoleh hasil yang lebih baik.

Jurnal Indonesia : Manajemen Informatika dan Komunikasi, Volume 5 No 1, Januari (2024), pp. 1019-1032
DOI : https://doi.org/10.35870/jimik.v5i1.582

5. Ucapan Terima Kasih

Ucapan terima kasih yang sebesar-besarnya penulis ucapkan kepada Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana, rekan-rekan komunitas Linux dan *Open-Source* Indonesia, tim riset, serta keluarga atas dukungannya selama penelitian berlangsung.

6. Daftar Pustaka

[1] Fujianto, A., & Waspada, I. (2016). Rancang Bangun Sistem Informasi Pengelolaan Dns Secara Terpusat (Studi Kasus Cv. Surya Putra Perkasa). *Jurnal Ilmiah Infokam*, *12*(1). DOI: https://doi.org/10.53845/infokam.v12i1.94.

[2] Brahara, B., Syamsuar, D., & Kunang, Y. N. (2020). Analysis of malware DNS attack on the network using domain name system indicators. Journal of Information Systems and Informatics, 2(1), 131-153. DOI: 10.33557/journalisi.v2i1.30.

[3] Umam, C., Handoko, L. B., & Rizqi, G. M. (2018). Implementation And Analysis High Availability Network File System Based Server Cluster. *Jurnal Transformatika*, *16*(1), 31-39. DOI: http://dx.doi.org/10.26623/transformatika.v16i1.841

[4] Septiantina, S., Kamil, A. L., & Solikhah, F. (2021). Teknik Failover Clustering Sebagai Solusi High Availability. *TEMATIK*, *8*(1), 104-109. DOI: https://doi.org/10.38204/tematik.v8i1.579.

[5] Rahmatulloh, A., & Firmansyah, M. S. N. (2017). Implementasi load balancing web server menggunakan haproxy dan sinkronisasi file pada sistem informasi akademik Universitas Siliwangi. *Jurnal Nasional Teknologi Dan Sistem Informasi*, *3*(2), 241-248. https://doi.org/10.25077/TEKNOSI.v3i2.2017.241-248.

[6] Rosalia, M. (2016). Implementation of High Availability Server Using Load Balancing and Failover Methods in Virtual Web Server Clusters. *e-Proceeding of Engineering*, *3*(3).

[7] Ramadhan, I., & Guarddin, G. (2022). Large-scale integrated infrastructure for asynchronous microservices architecture. *Jurnal Teknologi dan Sistem Komputer*.

[8] Sinlae, A. A. J., Bagir, M., & Prayitno, M. H. (2022). Analisis Perbandingan Algoritma Round-Robin dengan Least-Connection Terhadap Peningkatan Nilai Throughput Pada Layanan Web Server. *JURIKOM (Jurnal Riset Komputer)*, *9*(5), 1584-1590. DOI: http://dx.doi.org/10.30865/jurikom.v9i5.4995.

[9] Arta, Y. (2017). Penerapan Metode Round Robin Pada Jaringan Multihoming Di Computer Cluster. *IT Journal Research and Development*, *1*(2), 26-35. DOI: https://doi.org/10.25299/itjrd.2017.vol1(2).677.

[10] Fritsch, J., & Walker, C. (2012). CMQ-A lightweight, asynchronous high-performance messaging queue for the cloud. *Journal of Cloud Computing: Advances, Systems and Applications*, *1*, 1-13.

[11] Firmansyah, F., & Purnama, R. A. (2019). Filtering Domain Name Server (DNS) untuk Membangun Internet Sehat Menggunakan Routerboard Mikrotik. *JUITA: Jurnal Informatika*, *7*(1), 43-48. DOI: 10.30595/juita.v7i1.4164.

[12] Sitorus, S. P., & Zarlis, M. (2017). ANALISIS KINERJA NON CDN DAN GEO DNS PADA CDN MENGGUNAKAN NS-2. *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, *1*(2), 133-137. DOI: https://doi.org/10.30743/infotekjar.v1i2.79.

[13] Rodiah, R. (2020). IMPLEMENTASI HIGH AVAILABILITY UNTUK PENGURANGAN WAKTU DOWNTIME PADA JARINGAN DENGAN PROTOKOL HIGH AVAILABILITY FIRST HOP REDUDANCY PROTOCOL (FHRP). *Jurnal Ilmiah Informatika Komputer*, *25*(2), 147-159. DOI: http://dx.doi.org/10.35760/ik.2020.v25i2.2982.

[14] Thooriqoh, H. A., Azzmi, M. N., Tofan, Y. A., & Shiddiqi, A. M. Malicious Traffic Detection In Dns Infrastructure Using Decision Tree Algorithm. *Jurnal Ilmiah Teknologi Informasi*, *20*(1), 45-53. DOI: http://dx.doi.org/10.12962/j24068535.v19i3.a1054.

[15] Fauzi, A. (2020). Analisa Kinerja pada Standalone Server dan Clustering Server Teknologi RAC (Real Application Clustering) dengan Algoritma DNS (Domain Name System) Round Robin Berbasis Oracle Linux 6.4 di Lingkungan Virtual. *Jurnal Sistem Komputer*, *10*(2), 63-69.

[16] Riawati, A. D., Irfan, M., Khaeruddin, K., & Faruq, A. (2022). High Availability Dynamic Sharding Database Server Dengan Metode Fail Over Dan Clustering. *Jurnal Manajemen Informatika dan Sistem Informasi*, *5*(1), 1-10. DOI: https://doi.org/10.36595/misi.v5i1.416.

[17] Huda, A. N., & Susila, A. (2023). Infrastruktur Virtualisasi Data Center Berbasis Site Recovery Pada (PT. Informatics Oase). *OKTAL: Jurnal Ilmu Komputer dan Sains*, *2*(03), 823-832.

[18] Fitria, A. (2021). IMPLEMENTASI DOUBLE DEEP PACKET INTRUSION DETECTION DAN PREVENTION SYSTEM (IDPS) INSPECTION DENGAN PERANGKAT FIREWALL NGFW DAN APPLICATION SECURITY MANAGER PADA CLOUD DATACENTER PT. XYZ. *UG Journal*, *14*(6).

[19] Mukti, F. S., & Nugroho, F. E. (2023). Upaya Pencapaian Status High Availability Server Menggunakan Metode Load Balancing Berbasis Klaster pada Database Server PT. XYZ. *JATISI (Jurnal Teknik Informatika dan Sistem Informasi)*, *10*(1). DOI: https://doi.org/10.35957/jatisi.v10i1.5886.

[20] Iryani, N., Ayatri, K. D., & Wahyuningrum, R. D. (2022). Analisis performansi high availability cluster server menggunakan heartbeat pada private cloud: Performance analysis of high availability cluster servers using heartbeat on private cloud. *JITEL (Jurnal Ilmiah Telekomunikasi, Elektronika, dan Listrik Tenaga)*, *2*(2), 129-138. DOI: https://doi.org/10.35313/jitel.v2.i2.2022.129-138.

[21] Subekti, Z. M., Subandri, S., & Rakasiwi, G. (2019). PERANCANGAN INFRASTRUKTUR WEB SERVER DAN DATABASE MENGGUNAKAN METODE REPLICATION MIRROR DAN FAILOVER CLUSTERING. *Jurnal Cendikia*, *18*(1), 359-370.

[22] Pangestu, T., & Liza, R. (2022). Analisis Keamanan Jaringan pada Jaringan Wireless dari Serangan Man In The Middle Attack DNS Spoofing. *JiTEKH*, *10*(2), 60-67. DOI: https://doi.org/10.35447/jitekh.v10i2.571.

[23] Riskawati, R., & Marisda, D. H. (2020). The effectiveness of experimental method in teaching motion topic at senior high school level. *Jurnal Pendidikan Fisika*, *8*(1), 33-42. DOI: https://doi.org/10.26618/jpf.v8i1.3004.

[24] Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 45-77.

[25] Pratama, I. P. A. E. (2022). Design And Implementation Of An Artificial Intelligence-Based Heart Disease Diagnosis System. *Indonesian Journal of Engineering and Science*, *3*(1), 033-040. DOI: https://doi.org/10.51630/ijes.v3i1.33.