JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 9, No. 1, JANUARI, 2024, Pp. 50 – 58
ISSN: 2527 – 5836 (print) | 2528 – 0074 (online)

Server *Redundancy*: Performa Jaringan Mengunakan DNS *Failover* MikroTik pada Kasus *Private Server* dan *Public Server*

Tommi Alfian Armawan Sandi [(1)*], Firmansyah [(2)], Ahmad Fauzi [(3)] , Hanggoro Aji Al Kautsar [(3)]
[1,2,4] Informatika, Fakultas Teknik dan Informatika, Universitas Bina Sarana Informatika, Jakarta
[3] Sistem Informasi, Fakultas Teknologi Informasi, Universitas Nusa Mandiri, Jakarta
e-mail : {tommi.taf,firmansyah.fmy,hanggoro.hgr}@ bsi.ac.id, ahmad.azy@nusamandiri.ac.id.
* Penulis korespondensi.
Artikel ini diajukan 26 Oktober 2023, direvisi 14 Desember 2023, diterima 15 Desember 2023, dan dipublikasikan 25 Januari 2024.

Abstract

The digitization of user services has increased, in line with the need for VPS and cloud computing services, which are rampant among application and platform developers. Quite a few companies that create applications or application users have servers to handle user needs. Testing is carried out using the ICMP Protocol to get real-time results and can be measured. From Scenario 1, carrying out 20 test requests, we get a packet loss of 5% with RTT Avarage of 195,838ms and Mdev 4,103ms. If you apply DNS failover in scenario 2, the client will likely access the web a little slower, as evidenced by the packet loss being 25% greater in value. Compared to scenario 1, having a high standard deviation (Mdev) of round-trip times is not desirable. This variation is also known as jitter. Increased jitter can cause a bad user experience, especially in real-time audio and video streaming applications. However, this is still understandable because it only has a 1-5 second effect on the service. Next, in scenario 3, we can see that private and public servers have relatively high gab with 0% packet loss, which has a small Mdev value of 0.309ms. Therefore, the DNS failover method is a solution for network administrators who have problems related to server migration between public servers and private servers so that services can run even if a server is maintaining or downlinking.

Keywords: Server Redundancy, DNS Failover, RTT, Mdev, Private Server, Public Server

Abstrak

Digitalisasi layanan pengguna mengalami peningkatan, selaras dengan kebutuhan VPS dan layanan *cloud computing* yang merajalela di kalangan *development* aplikasi dan *platfrom*. Tidak sedikit dari perusahaan yang membuat aplikasi atau pengguna aplikasi memiliki server untuk mengangani kebutuhan penguna. Pengujian dilakukan dengan *protocol* ICMP untuk mendapatkan hasil yang *real-time* dan dapat diukur. Dari skenario 1 melakukan 20 kali *request* pengujian didapatkan *packet loss* 5% dengan *avarage* RTT sebesar 195,838ms dan Mdev 4,103ms, jika menerapkan *failover* DNS pada skenario 2, *client* kemungkinan mengakses web sedikit lambat terbukti dari *packet loss* yang hilang sebesar 25% lebih besar nilainya dibanding skenario 1 dan memiliki standar deviasi (Mdev) *round-trip times* yang tinggi tidaklah diinginkan. Variasi ini juga dikenal sebagai *jitter*. *Jitter* yang tinggi dapat menyebabkan pengalaman pengguna yang buruk, terutama dalam aplikasi *real-time* seperti *streaming audio* dan video. Walaupun begitu hal ini masih dalam kondisi dapat dimaklumkan karna hanya 1-5 detik pengaruhnya terhadap layanan. Selanjutnya pada skenario 3, kita dapat melihat jika perbedaan *private server* dan *public server* memiliki gap yang cukup tinggi dengan 0% *packet loss* yang tentunya memiliki nilai Mdev yang kecil 0,309ms. Oleh karena itu, jika metode *failover* DNS pastinya menjadi solusi administrator jaringan yang memiliki masalah terkait migrasi server antara *public server* dan *private server* supaya layanannya bisa berjalan walaupun ada server yang *maintenace* atau *downlink*.

Kata Kunci: *Server Redundancy,* DNS *Failover,* RTT, Mdev*, Private Server, Public Server*

JISKA (Jurnal Informatika Sunan Kalijaga)
Vol. 9, No. 1, JANUARI, 2024: 50 – 58

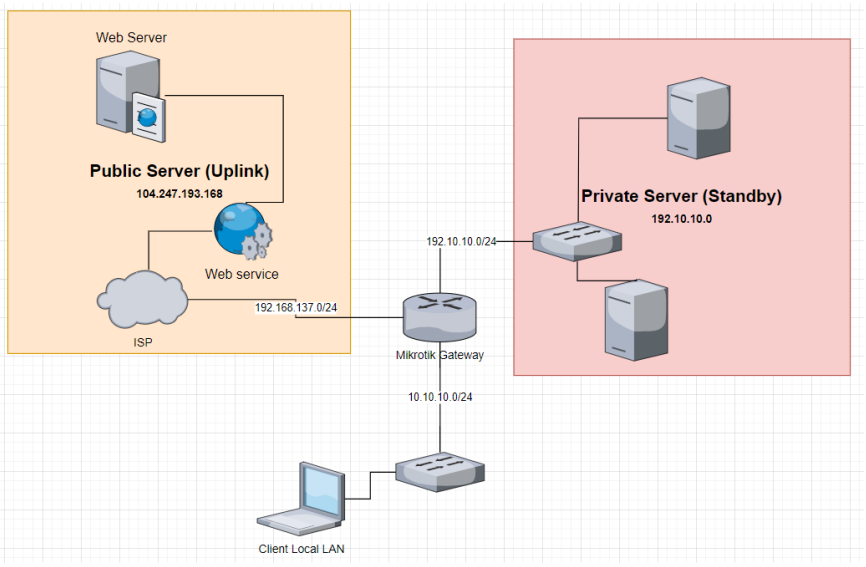
1. PENDAHULUAN

Digitalisasi layanan pengguna mengalami peningkatan, selaras dengan kebutuhan VPS dan layanan *cloud computing* (’Abidah et al., 2020) yang merajalela di kalangan *development* aplikasi dan *platfrom*. Tidak sedikit dari perusahaan yang membuat aplikasi atau pengguna aplikasi memiliki server untuk mengangani kebutuhan penguna. Terdapat 3 (tiga) model utama dalam komputasi awan yang dimanfaatkan untuk membangun sebuah *platfrom* di antaranya ada IaaS (*Infrastructure as a Service*), PaaS (*Platform as a Service*), dan SaaS (*Software as a Service*) (Nadeem, 2022). Pemanfaatan server telah meningkat secara signifikan karena virtualisasi sistem komputer, yang menyebabkan peningkatan konsumsi daya dan kebutuhan penyimpanan oleh pusat data dan dengan demikian juga meningkatkan biaya operasional (Alyas et al., 2022). Penyediaan IaaS penyedia *cloud* untuk menyediakan mesin virtual (VM) dan menggunakannya dengan cara yang mirip dengan klaster lokal (Malla & Christensen, 2020).

Fenomena migrasi server bukanlah hal yang baru dalam bidang ini (Hosseini Shirvani et al., 2020), walaupun banyaknya layanan yang diberikan oleh server ke pengguna membuat server mendapat perhatian khusus terlebih aksesbilitas dan performa (Bhardwaj & Rama Krishna, 2022). Manajemen migrasi pun tidak lagi menjadi bagian *system development* (Khan et al., 2022) namun juga bagian dari administrator jaringan. Seperti penelitian sebelumnya yang dilakukan mengenai proses migrasi *cloud computing* dari lingkungan Amazon EC2 ke VMware mengalami kendala yang terdapat di IaaS yang tidak maksimal dalam proses migrasi IaaS, konversi *image virtual machine* dari satu sistem ke sistem lain tidak menjamin keberhasilan proses migrasi (Jupriyadi et al., 2021; Wijayanto et al., 2021). Terdapat kecenderungan beberapa parameter seperti konfigurasi *central processing unit*, *random access memory*, jaringan, serta struktur aplikasi dan data yang mungkin bisa berubah dalam proses migrasi (Mafakhiri, 2019). Penelitian selanjutnya tentang implementasi *load balancing* dan *failover* pada proses migrasi *container* Docker menyebutkan ketersediaan web server yang tidak didukung dengan metode dapat menghambat kinerja dari suatu server, dikarenakan terjadinya penumpukan request pada server dapat membuat kegagalan jaringan (Ri et al., 2023; Rifiera & Nurwarsito, 2022). Pada analisis perbandingan layanan data server menggunakan *failover cluster* pada *platform* nginx dan apache menegaskan jika web server yang efektif untuk mendukung teknik *failover* serta perlu dibangun suatu *high availability web server cluster* (Marzuki et al., 2022) yang dapat menjalankan teknik *failover*, sehingga dapat meminimalisir resiko kegagalan *service* pada web server yang dapat menghambat efektifitas kerja penyedia layanan (Maila et al., 2020; Nannipieri et al., 2019). Pada saat ini dengan menjalankan *failover routing* tidak bisa memilah antara *traffic* ISP atau layanan dari server (Sandi et al., 2021). Oleh karena itu perlunya pengujian *failover* dengan DNS yang tersedia di masing-masing server untuk melakukan peralihan server tanpa mengurangi keandalan dari layanan tersebut (Dooley, n.d.; Rouse, 2013).

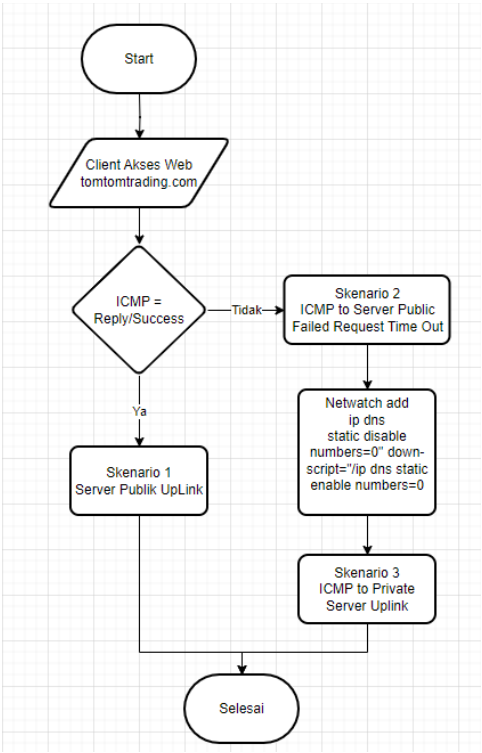
2. METODE PENELITIAN

Penelitian ini dilakukan dengan melakukan eksperimen data yang dipadukan oleh catatan observasi pada jaringan *Local Area Network* (Hae, 2021), perangkat yang digunakan dalam penelitian ini di antaranya: MikroTik dengan arsitektur x86 yang diunduh pada *virtual machine*, web server terbagi atas 2 (dua) yang terunduh pada VPS dan *private server* (*local*) pada setiap server menggunakan *operating system* Centos 7 dan terinstall CWP (Centos Web Panel/Control-WebPanel) *latest version*, serta *client* menggunakan perangkat laptop dan PC. Penelitian ini bersifat instrumental dan mengumpulkan semua data dengan menghasilkan data secara deskriptif dan menambilkan hasil server *redundancy* pada jaringan ketika mengalami *downlink*/*maintenance* server. Terlihat pada Gambar 1 merupakan skema jaringan yang digunakan pada penelitian server *redundancy*, performa jaringan menggunakan DNS *failover* MikroTik pada kasus *private server* dan *public server*.

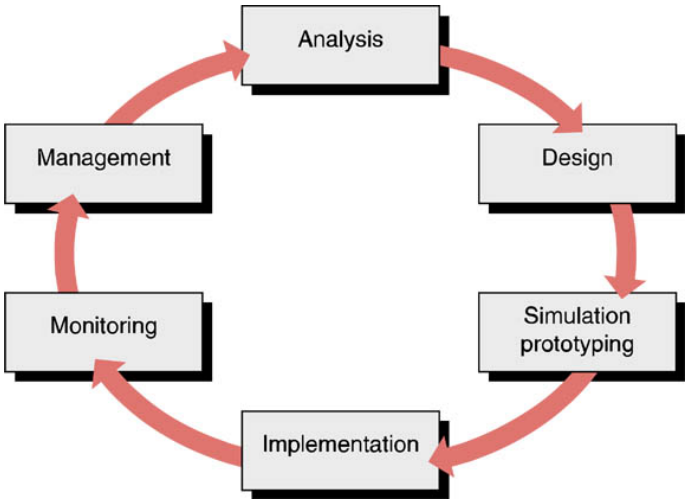


Gambar 1 Skema Jaringan

Tahapan pengujian yang akan digunakan dalam penelitian dapat dilihat pada Gambar 2. Diagram pengujian mengedepankan performa terhadap DNS *failover* terhadap kedua server tersebut. Detail pengujian di antaranya melakukan uji konektifitas terhadap *client* yang mengakses *public server* dalam keadaan normal, selanjutnya pengujian DNS *failover* dalam keadaan *public server*. Pada kondisi *downlink* kemudian dilakukan pengujian terakhir dengan *client* mengakses *private server* dalam keadaan normal. Pada implementasi *failover* ini berbeda dengan *failover* pada *routing*, parameter yang digunakan adalah *check gateway* dan ICMP. Dengan menggunakan ICMP dapat mengetahui secara *real-time* dan *responsive* terhadap konektivitas pada suatu perangkat (Ďulík et al., 2023).



Gambar 2 Diagram Alur Pengujian



Gambar 3 SPDLC

Dapat dilihat pada Gambar 3 merupakan metode penelitian yang digunakan dalam penelitian ini. Penelitian menggunakan metode *The Security Development Life Cycle (SPDLC)* yang memiliki 6 tahapan yaitu analisis, *design*, simulasi, implementasi, *monitoring*, dan manajemen (W & Fitriana, 2021).

1) Analisis yang diterapkan dengan mengidentifikasikan kebutuhan penguna dan permasalahan yang dihadapi sebelum melakukan tahapan berikutnya
2) *Design* dengan melakukan *re-map* (topologi) *Local Area Network* yang ada di lapangan dan disesuaikan dengan kebutuhan penguna.
3) Simulasi yang dilakukan pada penelitian ini dengan mengumpulkan kofigurasi-konfigurasi, mulai dari konfigurasi MikroTik maupun server dan dijalankan menggunakan *virtual box*.
4) Implementasi merupakan tahapan di mana penulis melakukan penerapan pada perangkat *rill* yang disesuaikan pada kebutuhan penguna, baik itu instalasi di sisi server maupun *router*.
5) *Monitoring* pada penelitian ini yaitu dengan melihat *log activity* pada MikroTik apakah jaringan telah berjalan dengan semestinya dan *failover* DNS berjalan dengan baik.
6) Manajemen melakukan evaluasi terhadap layanan jaringan yang telah diterapkan sesuai dengan tujuan awal penelitian.

3. HASIL DAN PEMBAHASAN

Skenario pengujian server *redundancy* performa jaringan menggunakan DNS *failover* MikroTik pada kasus *private server* dan *public server* dilakukan dengan cara pembuktian uji konektifitas jaringan kantor yang normal dengan konfigurasi dasar *gateway* sampai DNS yang sesuai dengan ISP yang diberikan. Skenario 2 yaitu menghentikan layanan dari *public server* dan melihat dari *client* sebelum dan setelah menggunakan *failover* ke *private server*. Skenario 3 adalah melakukan redundansi dari *private server* ke *public server* dan mengetahui *quality of service* terhadap layanan-layanan tersebut. Untuk mengimplenetasikan layanan jaringan menggunakan DNS *failover* pada MikroTik peneliti menggunakan spesifikasi IP *address* pada Tabel 1.

Tabel 1 IP *Address*

Perangkat	Interface	IP Address	Gateway
Router 1	Ether 1	192.168.137.2/24	192.168.137.1
	Ether 2	192.10.10.1/24	
	Ether 3	10.10.10.1/24	
Public server	NIC	104.247.193.168	
Private server	NIC	192.10.10.2	192.10.10.1
PC *Client* (1…24)	NIC	10.10.10.xxx/24	

3.1 Implementasi Server *Redundancy* dengan DNS *Failover*

Pada tahapan ini setiap alamat jaringan pada server harus diketahui dan diperlukan IP *static*. Hal ini dilakukan untuk mempermudah *router* mencari *route* mana yang harus dituju dengan memanfaatkan IP tersebut. Pada impementasi *failover* ini berbeda dengan *failover* pada *routing*, parameter yang digunakan adalah *check gateway* dan ICMP. Berdasarkan topologi yang diimplementasi terdapat beberapa *client* dan server yang harus mendapatkan layanan dari *router* yang sebagai *gateway* onfigurasi. *Basic config* yang harus dilakukan di antaranya IP *address*, IP *route*/*gateway*, IP server DNS, dan *firewall* NAT.

3.1.1 Konfigurasi IP *Address*

Konfigurasi IP *address* yang digunakan dalam penelitian ini yaitu seperti pada Gambar 4. Pada konfigurasi tersebut menambahkan IP *Address* di masing-masing *interface*, dengan melihat tabel IP yang telah ditetapkan.

```
[admin@MikroTik] >ip address add address=192.168.137.2/24 interface=ether1
[admin@MikroTik] >ip address add address=192.10.10.1/24 interface=ether2
[admin@MikroTik] >ip address add address=10.10.10.1/24 interface=ether3
[admin@MikroTik] > ip address print
Columns: ADDRESS, NETWORK, INTERFACE
#     ADDRESS          NETWORK       INTERFACE
0     192.168.137.2/24 192.168.137.0 ether1
1     192.10.10.1/24   192.10.10.0   ether2
2     10.10.10.1/24    10.10.10.0    ether3
```

Gambar 4 Konfigurasi IP *Address*

3.1.2 Konfigurasi DNS Server

Pada konfigurasi DNS masih menunggu semua IP server yang digunakan *public server* meliliki IP 104.247.193.168 dan *private server* 192.10.10.2 dimasukkan dalam DNS *list* dengan konfigurasi seperti pada Gambar 5. Selain itu, perlu ditambahkan IP server pada DNS *static*, Hasil dari DNS server bisa menentukan ke mana *client* dapat mendapatkan layanannya. Konfigurasi yang dilakukan seperti pada Gambar 6. Pada Gambar 7, Menunjukkan bahwa ada 2 DNS *static* yang berhasil di input dengan *private server* dan *public server*.

```
[admin@MikroTik] > ip dns set servers=8.8.8.8 allow-remote-request=yes
```

Gambar 5 Konfigurasi DNS Server

```
[admin@MikroTik] > ip dns static add address=192.10.10.2 name=tomtomtrading.com type=A ttl=1d
[admin@MikroTik] > ip dns static add address=104.247.193.168 name=tomtomtrading.com type=A ttl=1d
```

Gambar 6 Konfigurasi DNS *Static*

```
[admin@MikroTik] > ip dns static/print
Flags: X - DISABLED
Columns: NAME, ADDRESS, TTL
#   NAME                ADDRESS          TTL
;;; ServerPrivate
0 X tomtomtrading.com   192.10.10.2      1d
;;; Server Public
1   tomtomtrading.com   104.247.193.168  1d
[admin@MikroTik] >
```

Gambar 7 DNS *Static Print*

3.1.3 Konfigurasi *Route*

Konfigurasi *route* yang digunakan dalam penelitian ini yaitu seperti pada Gambar 8.

```
[admin@MikroTik] > ip route add gateway=192.168.137.1 distance=1
```

Gambar 8 Konfigurasi *Route*

3.1.4 Konfigurasi NAT

Konfigurasi NAT yang digunakan dalam penelitian ini yaitu seperti pada Gambar 9.

```
[admin@MikroTik] > ip firewall nat add chain=srcnat out-interface=ether1 action=masquerade
```

Gambar 9 Konfigurasi NAT

3.1.5 Konfigurasi Netwatch

Netwatch difungsikan untuk memonitor kondisi *host*. Hal ini dikarenakan setiap server kemungkinan tidak dapat dimonitor *real-time* oleh administrator jaringan. Dengan memanfaatkan Netwatch dapat melakukan tindakan pencegahan dan keputusan kondisi yang diperlukan. Konfigurasi Netwatch ditunjukkan pada Gambar 10. Pada Gambar 11 terdapat 2 konfigurasi yang bertugas untuk melakukan *ping reply* pada IP 104.247.193.168, jika kondisi IP tersebut *downlink* maka akan di alihkan otomatis ke *private server*.

```
[admin@MikroTik] > tool netwatch add host=104.247.193.168 type=simple interval=10m timeout=1 up-script="/ip dns static disable numbers=0" down-script="/ip dns static enable numbers=0"
[admin@MikroTik] > tool netwatch add host=104.247.193.168 type=simple interval=10m timeout=1 up-script="/ip dns static enable numbers=1" down-script="/ip dns static disable numbers=1"
```

Gambar 10 Konfigurasi Netwatch

```
[admin@MikroTik] > tool netwatch print
Flags: X - DISABLED
Columns: TYPE, HOST, TIMEOUT, INTERVAL, STATUS, SINCE
#    TYPE    HOST             TIMEOUT  INTERVAL  STATUS    SINCE
0    icmp    104.247.193.168                     down      oct/23/2023 13:23:26
1 X simple   192.50.10.2      1s       10s       unknown
[admin@MikroTik] >
```

Gambar 11 *Tool* Netwatch *Print*

3.2 Pengujian Konektifitas Skenario 1: *Client* Mengakses *Public Server* dalam Keadaan Normal

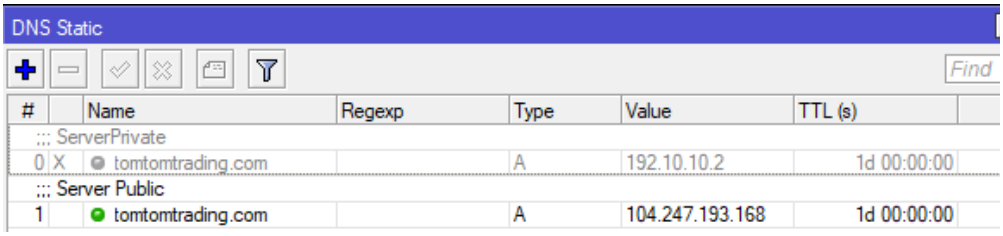
Skenario dalam uji yang pertama ialah melakukan tes terhadap layanan *public server* berupa tampilan web yang berisi informasi-informasi *trading*. *Protocol* ICMP dipilih lebih tepat digunakan dikarenakan fungsinya untuk mendiagnosis masalah komunikasi jaringan. Telihat pada Tabel 2 dan 3 perhitungan dengan melakukan 20 kali *request* layanan menggunakan *ping* pada *client*. Keadaan *public server uplink* terlihat pada Gambar 12.

Tabel 2 *Ping Statistics* Skenario 1

Packets		
Sent	*Received*	*Loss*
20	19	1 (5% *loss*)

Tabel 3 RTT (*Round Trip Times*) Skenario 1

Min	Avg	Max	Mdev
189,341 ms	195,838 ms	201,594 ms	4,103 ms



Gambar 12 *Public Server Uplink*

3.3 Pengujian Konektifitas Skenario 2: DNS *Failover* dalam Keadaan *Public Server Downlink*

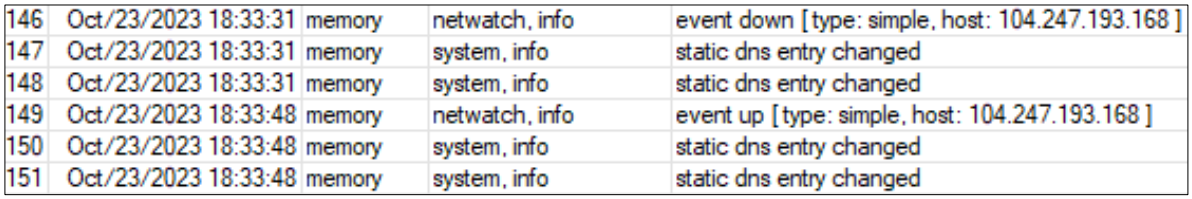
Skenario dalam uji yang kedua ialah melakukan tes terhadap layanan *public server* dengan menonaktifkan layanan atau 503 *Service Unavailable* dengan metode tes *ping* ICMP ke domain tomtomtrading.com dan dilakukan 20 kali *request* yang hasilnya terlihat pada Tabel 4 dan 5. Pada proses ini juga terjadi *failover* DNS yang terlihat pada Gambar 13. Waktu peralihan atau redundansi antara *public server* dan *private server* berlangsung 17 detik dari total 20 detik *request*. Perubahan *private server uplink* terlihat pada Gambar 14.

Tabel 4 *Ping Statistics* Skenario 2

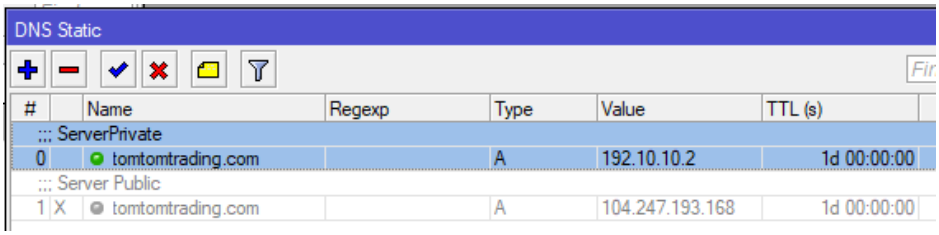
Packets		
Sent	*Received*	*Loss*
20	15	5 (25% *loss*)

Tabel 5 RTT (*Round Trip Times*) Skenario 2

Min	Avg	Max	Mdev
190,995 ms	205,844 ms	245,656 ms	13,160 ms



Gambar 13 *Failover* DNS *Success*



Gambar 14 Perubahan *Private Server Uplink*

3.4 Pengujian Konktifikas Skenario 3: Client Mengakses *Private Server* dalam Keadaan Normal

Skenario dalam uji yang terakhir ialah melakukan tes terhadap layanan *private server* yang letaknya dalam satu area yang terjangkau LAN, pengetesan menggunakan metode *ping* ICMP *private server* dengan 20 kali *request* terhadap layanan tersebut. Hasil pengujian terdapat pada Tabel 6 dan 7.

Tabel 6 *Ping Statistics* Skenario 3

Packets		
Sent	*Received*	*Loss*
20	20	0 (0% loss)

Tabel 7 RTT (*Round Trip Times*) Skenario 3

Min	Avg	Max	Mdev
2,154 ms	2,741 ms	3,100 ms	0,309ms

4. KESIMPULAN

Dengan melakukan beberapa uji coba dan eksperimen terhadap layanan pada *public server* dan *private* dapat disimpulkan dengan menggunakan *protocol* ICMP 20 kali *request* dari data skenario 1 pengujian didapatkan *packet loss* 5% dengan *avarage* RTT sebesar 195,838ms dan Mdev 4,103ms. Jika menerapkan *failover* DNS pada skenario 2, *client* kemungkinan mengakses web sedikit lambat. Terbukti dari *packet loss* yang hilang sebesar 25% lebih besar nilainya dibanding skenario 1 dan memiliki standar deviasi (Mdev) *round-trip times* yang tinggi tidaklah diinginkan. Variasi ini juga dikenal sebagai *jitter*. *Jitter* yang tinggi dapat menyebabkan pengalaman pengguna yang buruk, terutama dalam aplikasi *real-time* seperti *streaming* audio dan video. Walaupun begitu hal ini masih dalam kondisi dapat dimaklumkan karna hanya 1-5 detik pengaruhnya terhadap layanan. Selanjutnya pada skenario 3, kita dapat melihat jika perbedaan *private server* dan *public server* memiliki gap yang cukup tinggi dengan 0% *packet loss* yang tentunya memiliki nilai Mdev yang kecil 0,309ms. Oleh karena itu, jika metode *failover* DNS pastinya menjadi solusi administrator jaringan yang memiliki masalah terkait migrasi server antara *public server* dan *private server* supaya layanannya bisa berjalan walaupun ada server yang *maintenance* atau *downlink*.

DAFTAR PUSTAKA

'Abidah, I. N., Hamdani, M. A., & Amrozi, Y. (2020). Implementasi Sistem Basis Data Cloud Computing pada Sektor Pendidikan. *KELUWIH: Jurnal Sains Dan Teknologi*, *1*(2), 77–84. https://doi.org/10.24123/saintek.v1i2.2868

Alyas, T., Javed, I., Namoun, A., Tufail, A., Alshmrany, S., & Tabassum, N. (2022). Live Migration of Virtual Machines Using a Mamdani Fuzzy Inference System. *Computers, Materials & Continua*, *71*(2), 3019–3033. https://doi.org/10.32604/cmc.2022.019836

Bhardwaj, A., & Rama Krishna, C. (2022). A Container-Based Technique to Improve Virtual Machine Migration in Cloud Computing. *IETE Journal of Research*, *68*(1), 401–416. https://doi.org/10.1080/03772063.2019.1605848

Dooley, K. (n.d.). *What is Network Redundancy & Why is It Important?* Auvik. Retrieved January 25, 2024, from https://www.auvik.com/franklyit/blog/simple-network-redundancy/

Ďulík, M., Harakaľ, M., & Javurek, M. (2023). Advanced Methods for Network Infrastructure Analysis and Security. *2023 Communication and Information Technologies (KIT)*, 1–7. https://doi.org/10.1109/KIT59097.2023.10297110

Hae, Y. (2021). ANALISIS KEAMANAN JARINGAN PADA WEB DARI SERANGAN SNIFFING DENGAN METODE EKSPERIMEN. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *8*(4), 2095–2105. https://doi.org/10.35957/jatisi.v8i4.1196

Hosseini Shirvani, M., Rahmani, A. M., & Sahafi, A. (2020). A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter:

Taxonomy and challenges. *Journal of King Saud University - Computer and Information Sciences*, *32*(3), 267–286. https://doi.org/10.1016/j.jksuci.2018.07.001

Jupriyadi, J., Hijriyanto, B., & Ulum, F. (2021). Komparasi Mod Evasive dan DDoS Deflate Untuk Mitigasi Serangan Slow Post. *Techno.Com*, *20*(1), 59–68. https://doi.org/10.33633/tc.v20i1.4116

Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2022). Systematic Literature Review on Security Risks and its Practices in Secure Software Development. *IEEE Access*, *10*, 5456–5481. https://doi.org/10.1109/ACCESS.2022.3140181

Mafakhiri, J. (2019). Proses Migrasi Cloud Computing Dari Lingkungan Amazon Ec2 Ke Vmware. *Jurnal Bangkit Indonesia*, *8*(1), 1. https://doi.org/10.52771/bangkitindonesia.v8i1.85

Maila, H. H., Indra, D., & Satra, R. (2020). Analisis Perbandingan Layanan Data Server Menggunakan Failover Cluster pada Platform Nginx dan Apache. *Buletin Sistem Informasi Dan Teknologi Islam*, *1*(2), 87–91. https://doi.org/10.33096/busiti.v1i2.829

Malla, S., & Christensen, K. (2020). HPC in the cloud: Performance comparison of function as a service (FaaS) vs infrastructure as a service (IaaS). *Internet Technology Letters*, *3*(1), e137. https://doi.org/10.1002/itl2.137

Marzuki, K., Hanif, N., & Hariyadi, I. P. (2022). Application of Domain Keys Identified Mail, Sender Policy Framework, Anti-Spam, and Anti-Virus: The Analysis on Mail Servers. *International Journal of Electronics and Communications Systems*, *2*(2), 65–73. https://doi.org/10.24042/ijecs.v2i2.13543

Nadeem, F. (2022). Evaluating and Ranking Cloud IaaS, PaaS and SaaS Models Based on Functional and Non-Functional Key Performance Indicators. *IEEE Access*, *10*, 63245–63257. https://doi.org/10.1109/ACCESS.2022.3182688

Nannipieri, L., Cacciaguerra, S., Mirenna, S., Locati, M., Marletta, M., & Gucciardi, E. (2019). Making Linked Data more reliable with a failover server system: a case study with seismological data at INGV. *Annals of Geophysics*, *62*(Vol 62 (2019)), DM567. https://doi.org/10.4401/ag-8050

Ri, O.-C., Kim, Y.-J., & Jong, Y.-J. (2023). *Hybrid load balancing method with failover capability in server cluster using SDN*. http://arxiv.org/abs/2307.05552

Rifiera, S. N., & Nurwarsito, H. (2022). Implementasi Load Balancing dan Failover pada Proses Migrasi Container Docker. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *6*(5), 2025–2033. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/10967

Rouse, M. (2013, April 15). *Network Redundancy*. Techopedia. https://www.techopedia.com/definition/29305/network-redundancy

Sandi, T. A. A., Heristian, S., & Leksono, I. N. (2021). OPTIMALISASI FAILOVER DENGAN NETWATCH PADA MIKROTIK. *CONTEN : Computer and Network Technology*, *1*(1), 23–30. https://doi.org/10.31294/conten.v1i1.388

Wijayanto, D., Firdonsyah, A., Adhinata, F. D., & Jayadi, A. (2021). Rancang Bangung Private Server Menggunakan Platform Proxmox dengan Studi Kasus: PT.MKNT. *Journal ICTEE*, *2*(2), 41. https://doi.org/10.33365/jictee.v2i2.1333

W, Y., & Fitriana, Y. B. (2021). Analisis Network Security Komputer Tingkat Desa Menggunakan Metode Security Policy Development Life Cycle (SPDLC). *Jurnal Teknik Juara Aktif Global Optimis*, *1*(2), 11–21. https://doi.org/10.53620/jtg.v1i2.28