Jurnal Teknik Informatika (JUTIF)
Vol. 4, No. 4, August 2023, pp. 751-758

DOI: https://doi.org/10.52436/1.jutif.2023.4.4.691
p-ISSN: 2723-3863
e-ISSN: 2723-3871

NETWORK PROGRAMMABILITY FOR NETWORK ISSUE USING PARAMIKO LIBRARY

Dwi Ayu Mutiara[1], Khairunnisak Nur Isnaini[*2], Didit Suhartono[3]

[1]Information Technology, Faculty of Computer Science, Universitas Amikom Purwokerto, Indonesia
[2,3]Informatics, Faculty of Computer Science, Universitas Amikom Purwokerto, Indonesia

Email: [1]dwiayumutiara270@gmail.com, [2]nisak@amikompurwokerto.ac.id, [3]didit@amikompurwokerto.ac.id

(Article Received: November 16, 2022; Revision: Desember 2, 2022; Published: August 18, 2023)

Abstract

In a company, information technology is needed, especially computer networks, to facilitate data communication. The management of a computer network, of course, requires good administration. The criteria for whether or not a network is good can be seen from the performance, reliability, and security indicators so that it will not cause network issues. Events such as server downs, data loss, lost connections, and undetected computers cause the organization's business performance to be disrupted. This study's purpose is to detect network issues with network programmability technology automatically. Paramiko library supports network automation systems and implements OSPF routing protocol in finding the shortest path to send network packets. This study uses the PPDIOO flow, namely prepare, plan, design, implement, operate, and optimize, because it is considered by the flow of making network detection tools. The results showed that the design and implementation of a small-scale network were successfully built by utilizing network programmability technology and the paramiko library, which helps detect network conditions at any time. This design has a dashboard, provisioning, assurance, and policy features that allow administrators to manage and monitor information on each network device. The network design is fitted with REST-API technology and security through a secure shell (ssh) from the Network Controller that can detect the device's connection conditions and the device's health and update the DNS configuration used. Network Issues that have been seen are devices being down, and the connection being lost. Future research can improve features for network troubleshooting when the connection is lost.

Keywords*: paramiko library, network issue, network programmability, OSPF, PPDIOO*

NETWORK PROGRAMMABILITY UNTUK DETEKSI MASALAH JARINGAN MENGGUNAKAN *LIBRARY PARAMIKO*

Abstrak

Pada sebuah perusahaan kebutuhan akan teknologi informasi sangat dibutuhkan terutama jaringan komputer untuk memperlancar komunikasi data. Dalam pengelolaan sebuah jaringan komputer tentu memerlukan administrasi yang baik. Kriteria baik atau tidaknya sebuah jaringan bisa dilihat dari indikator *performance, reliability*, dan keamanannya sehingga tidak akan menimbulkan *network issue*. Hal yang saat ini terjadi seperti *server down, data lost,* koneksi terputus, dan komputer tidak terdeteksi menyebabkan kinerja bisnis organisasi terganggu. Tujuan penelitian ini adalah mendeteksi *network issue* secara otomatis dengan teknologi *network programmability. Library paramiko* digunakan untuk mendukung sistem otomatisasi jaringan dan menerapkan *routing protocol OSPF* dalam mencari jalur terpendek untuk mengirimkan paket jaringan. Penelitian ini menggunakan alur *PPDIOO* yaitu *prepare, plan, design, implement, operate,* dan *optimize* karena dianggap sesuai dengan alur pembuatan perancangan alat deteksi jaringan. Hasil penelitian menunjukan bahwa desain dan implementasi jaringan skala kecil berhasil dibangun dengan memanfaatkan teknologi *network programmability* dan *library paramiko* yang membantu mendeteksi kondisi jaringan setiap waktu. Desain ini dilengkapi dengan fitur *dashboard, provisioning, assurance,* dan *policy* yang membantu *administrator* untuk mengatur dan memonitoring informasi pada setiap perangkat jaringan. Desain jaringan dilengkapi dengan teknologi *REST-API* dan keamanan melalui *secure shell (ssh)* dari *Network Controller* ini mampu mendeteksi kondisi koneksi setiap perangkat, kesehatan setiap *device*, dan meng-*update* konfigurasi *DNS* yang digunakan. *Network Issue* yang berhasil dideteksi yaitu *device down* dan koneksi terputus. Penelitian selanjutnya dapat meningkatkan fitur untuk *troubleshooting jaringan* ketika koneksi terputus.

Kata kunci: *library paramiko*, masalah jaringan, *network programmability*, *OSPF*, PPDIOO

1. PENDAHULUAN

Teknologi saat ini berkembang sangat pesat, sehingga kebutuhan akan jaringan juga semakin meningkat [1]. Penggunaan jaringan komputer yang semakin tinggi dapat dilihat dari segala aspek kehidupan [2]. Komponen yang digunakan untuk membangun jaringan komputer, seperti *router, switch*, dan perangkat lain perlu dibangun dan dikonfigurasi dengan benar untuk menghindari masalah jaringan [3]. Koneksi jaringan komputer sangat penting karena jika terjadi masalah pada koneksi, maka berbagai aplikasi yang sedang berjalan tidak dapat digunakan [4].

Dalam sebuah organisasi komunikasi data merupakan hal yang sangat penting dan dapat berjalan melalui sebuah jaringan. Jaringan adalah sebuah media komunikasi antar *device* yang bisa bertukar informasi [5]. Pentingnya organisasi mempunyai jaringan yang baik yaitu agar proses bisnis dapat berjalan dengan lancar [6]. Kriteria baik atau tidaknya sebuah jaringan bisa dilihat dari indikator *performance*, *reliability*, dan keamanannya.

Saat ini jaringan di Universitas XY masih sering mengalami banyak *network issues*. *Network issues* yang sering terjadi antara lain *server down*, *data lost*, koneksi terputus, dan komputer tidak terdeteksi. *Server down* itu kerap kali terjadi saat pengajuan kartu rencana studi, kuliah online, dan penggunaan situs akademik secara serentak oleh mahasiswa sehingga menyebabkan koneksi terputus. Selain itu, *data lost* juga terjadi saat validasi presensi yang dilakukan oleh mahasiswa. Ketika *network issues* itu muncul, hingga saat ini *administrator* masih mengatasi masalah tersebut dengan melakukan pendekatan tradisional. Dalam pendekatan tradisional manajemen perangkat jaringan harus dikonfigurasi secara individual ke dalam setiap perangkat jaringan. Hal ini menyebabkan waktu menjadi tidak efesien dan banyak data *lost* yang tidak ter*backup* dengan baik [7].

Tantangan muncul ketika perangkat yang ditangani sangat banyak dan beragam [6]. Universitas XY tidak memiliki sistem terpadu yang dapat mendeteksi masalah jaringan. Jika Universitas XY saat ini masih menggunakan metode konvensional untuk mengatasi masalah tersebut, hal itu akan menghambat proses pembelajaran, penelitian dan pengabdian, layanan akademik, dan aktivitas secara umum. Tentunya semua hal tersebut dapat berpotensi merugikan proses bisnis di organisasi tersebut [8]**.**

Oleh karena itu, Universitas XY perlu meningkatkan performa dan fasilitas yang ada dengan cara menerapkan sistem kendali otomatis untuk mengatasi masalah jaringan. Sistem otomatisasi akan memudahkan *administrator* dalam mengelola jaringannya. Tugas konfigurasi jaringan seperti konfigurasi pencadangan, pemeliharaan perangkat, dan masalah jaringan dapat diotomatiskan oleh sistem yang akan dibuat sehingga waktu yang dihabiskan oleh *administrator* menjadi lebih hemat dan efesien.

Sistem otomatisasi yang akan dibangun memanfaatkan bahasa pemrograman python [9]. *Paramiko* merupakan salah satu *library* python yang dapat mendukung proses implementasi *network automation*. *Paramiko* dipilih karena proses *forwarding* lebih cepat dari *library netmiko* [10]. *Library paramiko* yang dimiliki oleh bahasa pemrograman *python* akan membantu sistem otomatisasi untuk mendapatkan informasi tentang kesehatan perangkat, koneksi perangkat, dan *network issues*.

Pada sistem otomatisasi akan menerapkan beberapa teknologi untuk menunjang perfoma desain jaringan yang dibangun antara lain *REST-API*, *Open Shorthest Path First (OSPF)*, dan *Secure Shell (SSH)*. *REST API* memungkinkan pengguna untuk mengakses *Network Controller* melalui *HTTP* [9]. *OSPF* adalah *routing protocol* yang mampu menentukan jarak terbaik dan terpendek ke setiap tujuan untuk mengirimkan paket jaringan [11] dan penggunaan *SSH* untuk mengatur keamanan rute dalam jaringan [12]. Selain itu penelitian ini akan menggunakan alur *PPDIOO* karena dianggap sebagai suatu pendekatan pengembangan sistem jaringan komputer yang bisa diterapkan dan berorientasi kepada area bisnis [13].

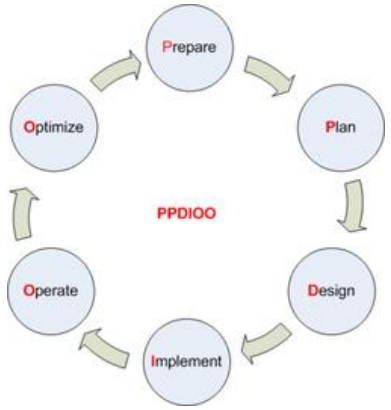
Penelian tentang pembuatan sistem otomatisasi yang dilakukan oleh [14] menghasilkan sistem keamanan jaringan yang mudah diatur oleh administrator untuk monitoring keamanan jaringan menggunakan Maltrail. Penelitian lain tentang penerapan otomatisasi jaringan oleh [15] menghasilkan *website* menggunakan *framework* Django dengan metode *waterfall* untuk melakukan konfigurasi *user* baru. Penelitian tentang penerapan *network programmability* yang dilakukan oleh [16] menghasilkan aplikasi untuk manajemen jaringan IOT dengan pendekatan Software Defined Network (SDN). Penelitian lain oleh [10] *library paramiko* dipilih dalam proses otomatisasi jaringan karena hanya melewati satu kali tahap eksekusi perintah sehingga waktu yang diperlukan akan jauh lebih cepat. Selain itu, penerapan *routing protocol OSPF* juga pernah dilakukan oleh [17] digunakan untuk mencari lintasan terpendek dalam melakukan *routing* yang menunjukan performansi dari *link state routing* berdasarkan parameter *Quality of Service (QOS)*. Penerapan *REST-API* juga digunakan oleh [9] pada *web service* data masyarakat yang menunjukan bahwa *REST-API* mampu mendukung kinerja *web service* agar bisa diakses melalui browser. Penelitian lain *secure shell (ssh)* dilakukan oleh [12] untuk melakukan *control* terhadap *server* yang mendukung fitur pengiriman file secara *SCP (Secure Copy)*. Metode *PPDIOO* juga digunakan oleh [13] untuk mendesain dan memvalidasi skema dari infrastruktur *server*.

Tujuan penelitian ini adalah membangun sebuah sistem otomatis untuk mendeteksi *network issue* pada Universitas XY dengan teknologi *network programmability*. Sistem otomatis ini menerapkan

library paramiko dan teknologi pendukung seperti *REST-API, Secure Shell (SSH), routing protocol OSPF*. Kontribusi pada penelitian ini adalah sistem otomatisasi yang dibangun mampu melakukan monitoring, deteksi informasi perangkat, deteksi *network issue*, dan *update DNS server.*

2. METODE PENELITIAN

Pendekatan penelitian yang digunakan pada penerapan *network programmability* jaringan skala kecil yaitu *PPDIOO* [18]. Tahapan-tahapan yang ada pada metode ini dapat dilihat pada gambar 1.



Gambar 1. tahapan penelitian

2.1 Prepare

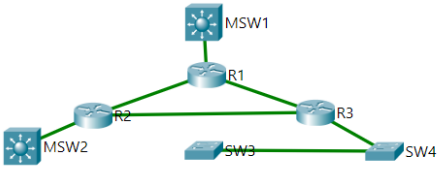
Pada tahap ini proses yang dilakukan adalah mengetahui dan mempersiapkan kebutuhan untuk jaringan awal yang akan dibangun. Melakukan identifikasi dan pengumpulan data yang berkaitan dengan kebutuhan jaringan di Universitas XY.

2.2 Plan

Pada tahap ini, penulis akan melakukan perencanaan jaringan yang akan dibuat berdasarkan kebutuhan serta menentukan *hardware* dan *software* yang akan mendukung proses penelitian. Sistem deteksi yang dibuat berupa website pada *localhost* dengan menggunakan port 58000.

2.3 Design

Dalam tahap ini proses yang dilakukan adalah membuat sebuah topologi jaringan serta melakukan proses *network programmability*. Topologi yang akan dibangun dapat dilihat pada gambar 2.



Gambar 1. rancangan topologi

Gambar 2 menunjukan bahwa topologi yang akan dibangun membutuhkan 3 *router*, 2 *switch* pada layer 3, dan 2 *switch* pada layer 2. Teknologi pendukung yang akan digunakan yaitu REST API untuk mendapatkan akses terhadap IP *network controller* sebagai perangkat pengontrol dalam jaringan yang dibangun. Library paramiko untuk membantu mendeteksi *network issue* yang terjadi pada jaringan. *Routing protocol OSPF* yang akan membantu pengiriman paket jaringan. *Secure shell (ssh)* untuk mengatur pengamanan dasar yang meliputi penerapan *password privilege, console dan remote access.*

2.4 Implement

Pada tahap implementasi ini, desain dan konfigurasi yang telah ditentukan akan diimplementasikan melalui simulator cisco packet tracer. Melalui cisco packet tracer desain tersebut akan menerapkan beberapa teknologi didalamnya. *Routing Protocol OSPF* untuk merutekan IP yang bekerja pada layer 3 OSI. *REST API* untuk mengakses *network controller* melalui *HTTP*. *Library paramiko* untuk mendukung sistem otomatisasi jaringan dalam memperoleh *service ticket, network health, dan network issues* dari *network controller*. *Secure Shell (SSH)* untuk mengatur keamanan rute dalam jaringan.

2.5 Operate

Langkah selanjutnya adalah melakukan *testing* pada sistem otomatis yang terhubung dengan *network controller*. *Testing* yang dilakukan adalah mencoba melakukan *login* dan fitur di setiap halaman *website. Web server* bisa diakses melalui http://controller.tiarabelajar.com atau localhost pada port 58000.

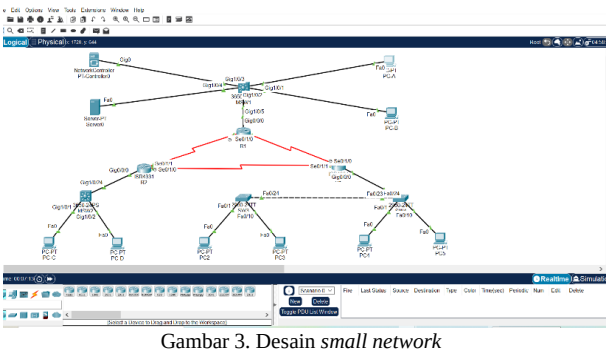
2.6 Optimize

Tahap terakhir yaitu meningkatkan sistem otomatisasi dengan mengoptimalkan sistem yang akan diterapkan sesuai dengan kebijakan–kebijakan yang dibuat oleh Universitas XY.

3. HASIL DAN PEMBAHASAN

3.1 Design

Layanan yang digunakan untuk membangun desain yang telah dirancang yaitu cisco packet tracer. Cisco Packet Tracer merupakan simulator yang menggunakan pendekatan 80% sesuai dengan perangkat aslinya [19]. Hasil rangkaian jaringan dapat ditunjukan pada gambar 3.



Gambar 3. Desain *small network*

Gambar 3 menunjukan desain yang dibangun berhasil *connected*. Selain itu, penerapan *REST-API*

untuk mendapatkan tiket juga berhasil diterapkan, hal ini terlihat pada gambar 4.

```
Starting network controller (Python)...
('Service Ticket: ', 'NC-89-e2d836bca5a94fca9d08-nbi')
network controller (Python) finished running.
```

Gambar 4. *Output REST API*

Pada gambar 4 terlihat bagian *output* yang ditandai dengan bingkai berwarna merah, *script* telah berhasil dieksekusi dan menampilkan informasi terkait tiket *REST API* untuk *access token* dari *network controller*. Penerapan *library paramiko* untuk membantu sistem otomatis dalam mendeteksi *network issue* dapat dilihat pada gambar 5.

```
Starting Project (Python)...
Persentase Network Health: 57%
Peringatan! Terjadi gangguan akses ke perangkat berikut:
-------------------------------------------------------------
NO. | PERANGKAT | WAKTU | DESKRIPSI
-------------------------------------------------------------
1. | PC5 | 2022-07-31 19:44:49 | Device is down
2. | SW3 | 2022-07-31 19:47:23 | Device is down
3. | SW4 | 2022-07-31 19:47:23 | Device is down
4. | R3 | 2022-07-31 19:47:23 | Device is down
5. | PC4 | 2022-07-31 19:47:39 | Device is down
6. | PC2 | 2022-07-31 19:47:39 | Device is down
7. | PC5 | 2022-07-31 19:47:39 | Device is down
8. | PC3 | 2022-07-31 19:47:39 | Device is down

Project (Python) error:
NameError: name 'roomId' is not defined in file main.py on line 57
```

Gambar 5. *Output library paramiko*

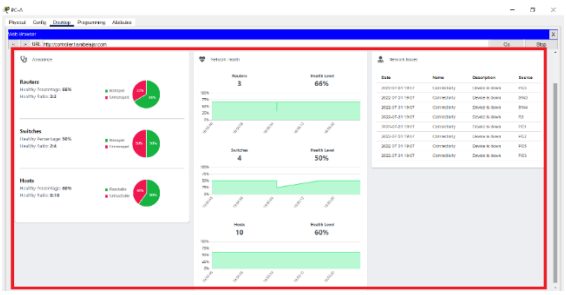
Gambar 5 bagian *output script* telah berhasil dieksekusi dan menampilkan informasi terkait *network issues (device is down)* yaitu gangguan akses ke 3 (tiga) perangkat meliputi *PC, switch dan router*. Informasi yang didapat yaitu peringatan ganguan akses perangkat. Selain itu sistem otomatisasi menerapkan teknologi pendukung lainnya yaitu *routing protocol OSPF* untuk mengatur kegiatan *routing* pada jaringan dan SSH untuk mengatur keamanan jalur jaringan.

3.2 Implementasi

Pada tahap implementasi *library paramiko*, desain dan konfigurasi berhasil diimplementasikan melalui simulator cisco packet tracer dengan menerapkan beberapa teknologi antara lain:

3.2.1 Penggunaan *Library Paramiko*

Bahasa pemrograman python dengan *library paramiko* digunakan untuk otomatisasi jaringan. *Script* python digunakan untuk memperoleh *service ticket*, *network health*, dan *network issues* di *network controller*. *Network issue* yang dapat dideteksi oleh sistem otomatisasi dilihat pada gambar 6.



Gambar 6 *Tampilan Network Issue*

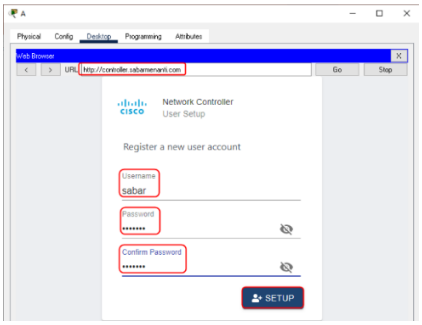
Gambar 6 yang ditandai dengan bingkai berwarna merah, *script* dengan *library paramiko* telah berhasil dieksekusi sehingga menampilkan informasi terkait *network issues* pada halaman *assurance* yaitu gangguan akses 4 *PC*, 2 *switch, dan* 1 *router*. Hal ini menyebabkan kondisi kesehatan perangkat menurun menjadi 57%.

3.2.2 *REST API*

REST API memungkinkan pengguna untuk mengakses *network controller* menggunakan antarmuka *REST* melalui *HTTP*. *REST API* ini akan meminta tiket yang mampu mengizinkan aplikasi lainnya mengakses *network controller* atau melalui *script* yang ditulis untuk mengotomatisasi jaringan.

3.2.4 *Secure Shell (SSH)*

Keamanan lalu lintas jaringan menggunakan *Secure Shell (SSH).* Gambar 7 merupakan *setting* dan tampilan dari konfigurasi keamanan jaringan yang dibangun.

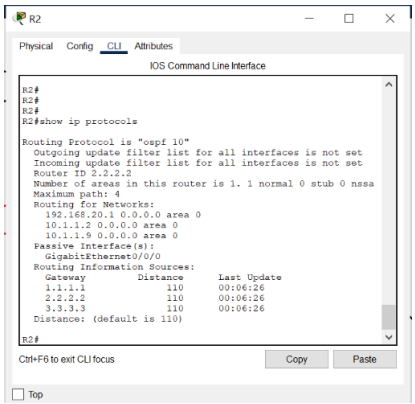


Gambar 7 *Secure Shell*

Gambar 7 menunjukan implementasi *SSH* berhasil dibangun dengan memerlukan *username* dan *password* untuk *login*.

3.2.5 *Routing Protocol OSPF*

Lalu lintas yang digunakan pada desain jaringan ini adalah *routing protocol OSPF*. Gambar 8 menunjukan implementasi *routing protocol OSPF* pada desain jaringan yang dibangun.



Gambar 8 Implementasi *routing protocol OSPF*

Gambar 8 merupakan implementasi *routing protocol OSPF*. Pada gambar tersebut IP router telah diatur pada area *0 (backbone)*. *OSPF* akan membantu sistem otomatis untuk mendeteksi *rute path* pada setiap perangkat.

3.3 Operate

Pada langkah ini akan dilakukan *testing* pada beberapa halaman yang dikontrol melalui perangkat *network controller*. Sistem otomatis yang dapat

mendeteksi *network issue* akan terlihat pada halaman *assurance.*

3.3.1 Halaman *Form Login*

Halaman utama pada sistem otomatisasi yaitu halaman *login.* Halaman *login* terdiri dari *username* dan *password*. Halaman ini berfungsi untuk memproteksi pengguna yang ingin mengakses sistem.

3.3.2 Halaman *Dashboard* Utama

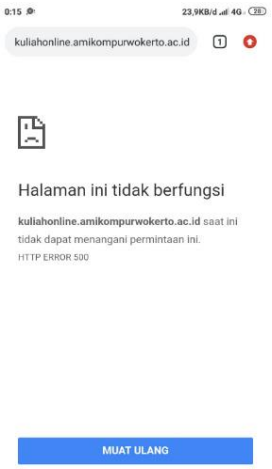
Halaman ini merupakan halaman untuk melakukan monitoring kondisi jaringan yang telah dibangun oleh *administrator* jaringan dengan beberapa menu seperti *prosentase* kesehatan koneksi, *prosentase* kesehatan perangkat, masalah jaringan, jumlah *host*, dan jumlah perangkat pada jaringan.

3.3.3 Halaman *Provisioning*

Halaman ini merupakan halaman untuk melakukan proses pembacaan perangkat mulai dari jenis perangkat, *IP address*, sampai status konfigurasi perangkat pada jaringan yang dibantu dengan *credential ssh* yang telah diatur pada saat konfigurasi diawal.

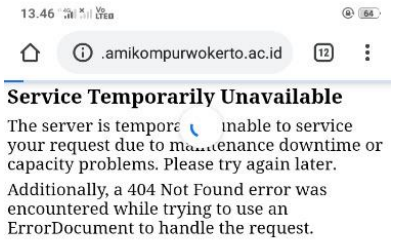
3.3.4 Halaman *Assurance*

Halaman ini merupakan halaman yang digunakan untuk melihat detail kesehatatan *device*, deteksi *network issue*, dan detail *host, topology*, serta *path trace (*rute paket). Pada Universitas XY telah terjadi masalah jaringan *server down* dengan kode 500 yang ditunjukan pada gambar 9.



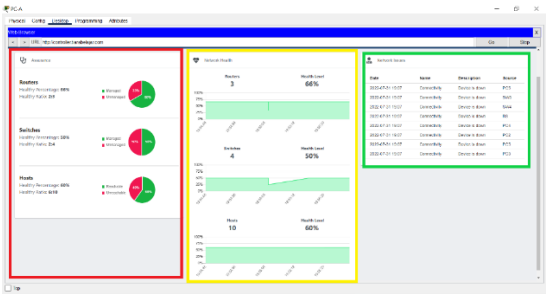
Gambar 9 Tampilan *server down* kode 500

Gambar 9 merupakan tampilan *network issue server down* yang pernah terjadi di Universitas XY. *Server down* tersebut menyebabkan situs kuliah *online* di Universitas XY tidak dapat diakses. Sehingga menyebabkan proses pembelajaran secara daring terhambat. Selain itu, masalah *server down* dengan kode 404 yang terjadi di Universitas XY dapat dilihat pada gambar 10.



Gambar 10. Tampilan *server down Kode 404*

Gambar 10 merupakan tampilan *server down* yang pernah terjadi di Universitas XY. *Server down* dengan kode 404 ini menunjukan bahwa *server* tidak dapat menemukan halaman atau situs yang dicari. Untuk mengatasi masalah tersebut, maka dibuatlah sistem otomatisasi untuk mendeteksi *network issue* pada jaringan. Pada gambar 11 dilakukan simulasi terdeteksinya *network issue* yang muncul yaitu perangkat *down.*



Gambar 11. Halaman *assurance*

Gambar 11 merupakan halaman yang menampilkan informasi lebih rinci mengenai kondisi koneksi perangkat dibandingkan dengan halaman *dashboard* utama, sehingga *administrator* akan mudah mengetahui *router*, *switch*, atau *host* mana yang mengalami masalah jaringan. Pada gambar yang berbingkai warna merah menampilkan rasio kesehatan perangkat yang terjadi ditunjukan dengan diagram lingkaran yang menginformasikan ada 2 *router*, 2 *switch*, dan 6 *host* yang kondisinya *up* (dalam kondisi hidup). Pada gambar yang berbingkai warna kuning menampilkan kesehatan jaringan yan menginformasikan kesehatan *router* 66%, *switch* 50%, dan *host* 60%. Kondisi kesehatan *router, switch, dan host* yang kurang dari 100% menyebabkan proses pengiriman paket jaringan antar perangkat terhambat sehingga berdampak pada aktivitas pertukaran data antar komputer tidak maksimal. Pada gambar berbingkai warna hijau menampilkan informasi detail *network issue* yang terjadi yaitu pada 4 *PC, 2 Switch dan 3 Router* yang mengalami kondisi *down* (dalam kondisi mati). Hal ini menunjukan bahwa sistem deteksi dapat digunakan pada kondisi sebenarnya seperti ketika *network issue* muncul karena sejalan dengan *network issue* yang terjadi di Universitas XY.

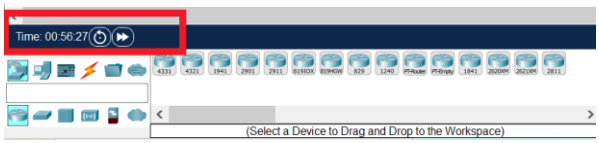
3.3.5 Halaman *Policy*

Halaman ini merupakan halaman yang digunakan untuk merubah pengaturan pada jaringan

dengan sekali *setup*. Pengaturan disini berfokus pada *DNS server*. Halaman ini diperlukan karena *DNS server* berfungsi untuk menyimpan semua *IP address* yang digunakan dalam *hostname*. Sehingga Fitur yang ada pada halaman ini mampu membantu sistem dalam mendeteksi *IP address* pada setiap perangkat.

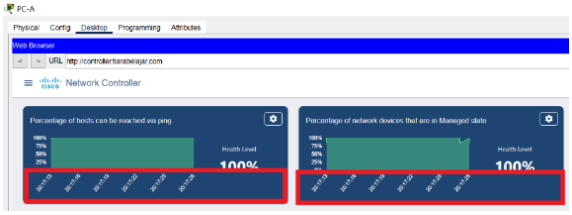
3.3.6 Perbandingan Sistem Otomatisasi dan Sistem Manual

Sistem otomatis dibuat untuk mempercepat waktu konfigurasi oleh *administrator*. Sistem ini berpotensi meningkatkan kinerja jaringan [20]. Sedangkan, pada sistem manual *administrator* perlu melakukan konfigurasi satu persatu pada setiap *router, switch* dan perangkat lainnya yang terhubung dalam jaringan. Waktu yang didapat pada sistem manual bisa dilihat pada gambar 12.



Gambar 12. Waktu konfigurasi perangkat

Gambar 12 yang ditandai dengan bingkai berwarna merah menunjukan waktu konfigurasi secara manual yaitu 56 Menit. Pembuktian waktu tersebut dilakukan secara simulasi melalui cisco packet tracer. Waktu yang diperlukan untuk konfigurasi secara otomatis dapat dilihat pada gambar 13.



Gambar 13. Waktu konfigurasi dan deteksi sistem otomatis

Gambar 13 yang ditandai dengan gambar berbingkai warna merah menunjukan rentang waktu yang diperlukan untuk melakukan konfigurasi dan deteksi kondisi perangkat. Analisis yang telah dilakukan pada kinerja sistem menunjukan bahwa sistem otomatis hanya membutuhkan waktu 0,3 detik-5 menit untuk dapat melakukan perubahan konfigurasi dan mendeteksi kondisi perangkat *down* atau *up*. Perbandingan waktu respon ditinjau menggunakan standar pembanding konfigurasi dan deteksi bisa dilihat pada tabel 2.

Tabel 2. Perbandingan waktu konfigurasi

No	Sistem	Waktu
1	Manual	56 Menit
2	Otomatis	0,3 detik – 4 Menit

Pada tabel 2 menunjukan bahwa waktu yang dibutuhkan untuk melakukan konfigurasi melalui sistem otomatis jauh lebih cepat dibandingkan menggunakan sistem manual. Hal ini tentu akan menghemat waktu *administrator* dalam melakukan konfigurasi jaringan. Sistem otomatis juga dapat mengurangi risiko kesalahan konfigurasi yang kerap kali terjadi.

4.4 *Optimize*

Dari hasil implementasi dan *operate* peningkatan yang perlu dilakukan adalah melakukan konfigurasi ulang agar memperoleh hasil yang lebih baik. Selain itu, Potensi gangguan terhadap sistem dapat diidentifikasi dengan mempertimbangkan hasil analisis pihak Universitas XY. Permintaan pemeliharaan jaringan akan ditetapkan [20]. Halaman *provisioning* sistem otomatis juga perlu pengoptimalan fitur *last update*.

4. DISKUSI

Desain rangkaian jaringan berhasil dibangun sesuai dengan topologinya. Desain jaringan tersebut menerapkan teknologi *network programmability* yang membangun sistem otomatisasi. Sitem otomatisasi ini dibangun dengan menerapkan beberapa teknologi seperti *library paramiko, REST-API, SSH,* dan *routing protocol OSPF*. *Library paramiko* diterapkan untuk menjalankan sistem otomatisasi sesuai dengan fungsingya. Fungsi yang dijalankan pada sistem otomatisasi adalah mendeteksi informasi perangkat, jumlah perangkat, kesehatan koneksi, kesehatan *device* dan masalah jaringan. Hal ini menunjukan bahwa *library paramiko* tepat digunakan dalam membuat sistem otomatisasi untuk mendeteksi masalah jaringan.

Sistem otomatisasi dibuat dalam bentuk *website*. Dalam hal ini pembuatan website pada *localhosting* tetap memerlukan sebuah pengujian. Pengujian ini dilakukan agar halaman pada website sistem otomatisasi dapat berjalan dengan baik dan meminimalisir risiko bug yang terjadi. *Testing* yang dilakukan pada sistem otomatisasi ini yaitu pengujian halaman *login*, halaman *dashboard* utama, halaman *assurance*, dan halaman *policy*. Pengujian deteksi *network issue* dilakukan pada halaman *assurance* karena informasi detail mengenai masalah jaringan terdapat pada halaman ini.

Setelah sistem otomatisasi ini dibangun, diimplementasi, dan diuji maka tahap selanjutnya yaitu optimalisasi. Optimalisasi tetap perlu dilakukan dengan cara menganalisis indikator hasil yaitu konfigurasi jaringan dan fitur pada sistem otomatisasi. Cara tersebut bertujuan untuk meningkatkan efektifitas fitur sistem otomatisasi agar semakin memudahkan administrator dalam mengelola jaringannya.

5. KESIMPULAN

Penerapan *network programmability* berhasil dibuat sehingga sistem secara otomatis dapat mendeteksi *network issue* antara lain *device down* dan koneksi terputus. Disamping itu desain jaringan dilengkapi dengan fitur *dashboard* yang menampilkan kesehatan *device* dan koneksi dalam bentuk prosentase, jumlah *device* pada jaringan yang dibangun, jumlah *host*, dan *network issue (device down)*. Fitur *provisioning* menampilkan detail *network device, credentials*, dan *discovery*. Pada fitur *assurance*

menampilkan detail *network issue*, kesehatan *hosts, router, dan switch*, tampilan *topology*, dan *path trace* (rute paket) dengan hasil kurang dari 70% *device* yang digunakan tidak dalam kategori bekerja secara optimal. Fitur *policy* yang digunakan untuk mengatur jaringan ketika akan dilakukan perubahan konfigurasi *DNS***.** Desain yang dilengkapi dengan teknologi *REST-API* ini mampu mempermudah *administrator* untuk mengakses sistem otomatis melalui *browser* yang keamanan jalurnya menerapkan *secure shell (ssh).* Desain ini dapat membantu menyelesaikan persoalan *administrator* untuk melakukan *monitoring* jaringan, sehingga ketika *user* mengalami interkoneksi, *administrator* dapat memperbaikinya dengan cepat. Hal ini tentu akan memperlancar kinerja bisnis di Unversitas XY.

DAFTAR PUSTAKA

[1] J. Sidabutar, "Desain Jaringan Komputer Terintegrasi Menggunakan Arsitektur Campus LAN," *J. Jaring SainTek*, vol. 2, no. 1, pp. 25–32, 2020, doi: 10.31599/jaring-saintek.v2i1.64.

[2] S. Nugroho, B. Pujiarto, U. M. Magelang, and P. Korespondensi, "Network Automation Pada Beberapa Perangkat Router Menggunakan Pemrograman Python," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 9, no. 1, pp. 79–86, 2022, doi: 10.25126/jtiik.202293947.

[3] R. Riski Amalia, Toibah Umi Kalsum, "Analisis dan Implementasi Software Defined Networking (SDN) untuk Automasi Perangkat Jaringan," *J. Inform. dan Teknol.*, vol. 4, no. 2, pp. 312–322, 2021.

[4] Kementrian ESDM, "Aturan Jaringan Sistem Tenaga Listrik (Grid Code)," 2020. [Online]. Available: https://jdih.esdm.go.id/storage/document/PM ESDM No 20 Tahun 2020.pdf

[5] M. A. S. Noris and R. Andrianto, "Jaringan Komputer," Medan, Sumatera Utara, 2020. doi: 10.13140/RG.2.2.25440.23042.

[6] M. Fahmi, M. Maisyaroh, I. Komarudin, S. Faizah, and I. Fadhilah, "Otomatisasi Jaringan Menggunakan Script Python Untuk Penyediaan Konfigurasi Internet Dan Manajemen Mikrotik," *Bina Insa. Ict J.*, vol. 8, no. 1, p. 53, 2021, doi: 10.51211/biict.v8i1.1517.

[7] W. Prabowo, "Rancang Bangun Automasi Jaringan Komputer Dengan Python Program Studi Teknik Informatika Politeknik Negeri Malang," Malang, 2021.

[8] R. A. Wiryawan and N. R. Rosyid, "Pengembangan Aplikasi Otomatisasi Administrasi Jaringan Berbasis Website Menggunakan Bahasa Pemrograman Python," *Simetris*, vol. 10, no. 2, pp. 1–12, 2019.

[9] M. I. Perkasa and E. B. Setiawan, "Pembangunan Web Service Data Masyarakat Menggunakan REST API dengan Access Token," *J. Ultim. Comput.*, vol. 10, no. 1, pp. 19–26, 2018, doi: 10.31937/sk.v10i1.838.

[10] K. Nugroho, A. D. Abrariansyah, and S. Ikhwan, "Perbandingan Kinerja Library Paramiko dan Netmiko dalam Proses Otomasi Jaringan," *InfoTekJar J. Nas. Inform. dan Teknol. Jar.*, vol. 5, no. 1, pp. 1–8, 2020.

[11] R. M. Negara and R. Tulloh, "Analisis Simulasi Penerapan Algoritma OSPF Menggunakan RouteFlow pada Jaringan Software Defined Network (SDN)," *J. Infotel*, vol. 9, no. 1, p. 75, 2017, [Online]. Available: http://ejournal.st3telkom.ac.id/index.php/infotel/article/view/172

[12] R. Pratama, M. Orisa, and F. Ariwibisono, "Aplikasi Monitoring Dan Controlling Server Menggunakan Protocol Icmp (Internet Control Message Protocol) Dan Ssh (Secure Shell) Berbasis Website," *JATI (Jurnal Mhs. Tek. Inform.*, vol. 4, no. 1, pp. 397–403, 2020, doi: 10.36040/jati.v4i1.2310.

[13] L. Hernandez and G. Jimenez, "Design and validation of a scheme of infrastructure of servers, under the PPDIOO methodology, in the university Institution-ITSA," *Adv. Intell. Syst. Comput.*, vol. 763, no. April, pp. 367–379, 2019, doi: 10.1007/978-3-319-91186-1_38.

[14] M. Anis, A. Hilmi, and E. Khujaemah, "Network Security Monitoring With Intrusion Detection System," *J. Tek. Inform.*, vol. 3, no. 2, pp. 249–253, 2022, [Online]. Available: https://doi.org/10.20884/1.jutif.2022.3.2.117

[15] Y. Chandra *et al.*, "Website Network Automation Design And Implementation In Rt Rw Perancangan Dan Implementasi Website Network Automation Pada Rt Rw Net Dusun Senden Magelang Dengan Framework," *J. Tek. Inform.*, vol. 3, no. 5, pp. 1313–1322, 2022.

[16] E. Municio, S. Latre, and J. M. Marquez-Barja, "Extending Network Programmability to the Things Overlay Using Distributed Industrial IoT Protocols," *IEEE Trans. Ind. Informatics*, vol. 17, no. 1, pp. 251–259, 2021, doi: 10.1109/TII.2020.2972613.

[17] I. N. Khoerotunisa, S. N. Hertiana, and R. M. Negara, "Analysis of User Mobility Performance on Software Defined Wireless Network Using Dijkstra Algorithm," *J. Tek. Inform.*, vol. 2, no. 2, pp. 127–133, 2021, doi: 10.20884/1.jutif.2021.2.2.84.

[18] R. N. Dasmen and Rasmila, "Rancang Bangun Vlan Pada Jaringan Komputer Rri Palembang Dengan Simulasi Cisco Packet Tracer," *J. Teknol.*, vol. Vol. 11 No, no. 1, pp. 47–56, 2019, [Online]. Available: https://jurnal.umj.ac.id/index.php/jurtek/article/view/2745

[19] SMK Wikrama 1, *Buku Cisco Paket Tracer*. Jepara, 2017.

[20] Kemenaker, "Keputusan Menteri Ketenagakerjaan Republik Indonesia Nomor 321 Tahun 2016 Tentang Penetapan Standar Kompetensi Kerja Nasional Indonesia Kategori Informasi dan Komunikasi Golongan Pokok Telekomunikasi Bidang Jaringan Komputer," Jakarta, 2016.