Implementasi DNS Dari Sudut Pandang Efek *Cache* dan *Resolver* Bersama Untuk Mengurangi Beban Pada Sistem

Fahmi Fauzan[1], Catur Wirawan Wijiutomo.[2], Muhammad Arief Nugroho[3]

[1,2,3] Universitas Telkom, Bandung
[1]fahmiif@students.telkomuniversity.ac.id,
[2]caturwijiutomo@telkomuniversity.ac.id,
[3]arief.nugroho@telkomuniversity.ac.id

Abstrak

Cara kerja kompleks dengan sistem yang sulit untuk dikelola dan dipecahkan karena terlalu banyak komponen yang berbeda membuat penyelesaian Shared Resolver menjadi suatu masalah bagi client. Beberapa penelitian mencari solusi dengan membuat server lokal sendiri yang dapat mempengaruhi kinerja jaringan serta mendapatkan jawaban atas permintaan client dari sisi waktu query yang didapatkan. Pada penelitian kali ini akan dilakukan suatu pembangunan server lokal sendiri dengan biaya yang murah untuk ukuran server tunggal namun dapat mempengaruhi kinerja jaringan menggunakan Embedded System Raspberry Pi sehingga dapat menjadi suatu penawaran yang baik bagi client. Pembangunan Server ini akan dilakukan pengujian dengan penggunaan Shared Resolver dan Resolver local full service sebagai server lokal pada Raspberry Pi untuk mengetahui hasil yang lebih baik. Didapatkan hasil bahwa pembuatan server lokal pada Raspberry Pi sebagai Resolver Local Full Service menjadi suatu solusi yang baik bagi client dengan menghemat waktu pencarian web server dari segi pengujian kinerja jaringan serta dapat menyimpannya ke cache agar dapat menyederhanakan sistem. Dan terakhir dari sisi pengujian filtering pada pembuatan server di Raspberry Pi sebagai Resolver local full-service dapat memblokir situs yang dianggap melanggar kebijakan.

Kata kunci : ***Shared Resolver, Resolver local full-service, Server Raspberry Pi***

Abstract

The complex way of working with a system that is difficult to manage and solve due to too many different components makes solving Shared Resolver a problem for the client. Several studies are looking for a solution by creating their own local server that can affect network performance and get answers to client requests from the query time obtained. In this study, a local server development will be carried out at a low cost for a single server size but can affect network performance using the Raspberry Pi Embedded System so that it can be a good offer for clients. The development of this server will be tested by using Shared Resolver and local full service Resolver as a local server on the Raspberry Pi to find out better results. The results show that making a local server on the Raspberry Pi as a Local Full Service Resolver is a good solution for the client by saving time on searching web servers in terms of network performance testing and being able to store it in the cache in order to simplify the system. And lastly, from the side of filtering testing on server creation on the Raspberry Pi as a full-service local resolver, it can block sites that are considered policy violations.

Keywords: ***Shared Resolver, Resolver local full-service, Server Raspberry Pi***

1. Pendahuluan

1.1. Latar belakang

Di era modern ini segala bentuk kegiatan yang dilakukan untuk memperoleh suatu pengetahuan dan informasi dapat diakses di *web browser*. Untuk mempermudah proses pencarian halaman *web server*, pengunggah atau penyedia halaman informasi melakukan Sistem Penamaan *Domain*. Ketika *client* mengetik nama *web* ke *browser*, komputer *client* perlu mengubahnya menjadi alamat IP sehingga dapat menghubungi *server web* itu dan mengirimkan halaman *web* kepada *client*. *Resolver DNS Server* merupakan peran kunci dalam mengubah tautan *web* ke dalam Alamat IP. Yang biasanya terjadi pada sistem kerja *Resolver* DNS *Server* selama ini yaitu terdapat pada *client* yang meminta sistem ISP untuk menghubungkan *client* kepada layanan informasi yang tersedia di *World Wide Web* atau WWW yang dijadikan sebagai suatu penyelesaian utama dengan mencari *server* keluar. Penyelesaian ini merupakan suatu penerapan dari cara kerja *server Shared Resolver* yang mana memecahkan masalah menjadi berbagai bagian kecil. Selain itu *Shared Resolver* terkadang membuat sistem sulit untuk dikelola dan dipecahkan karena cara kerja yang kompleks [1]. Untuk dapat terhubung ke suatu jaringan internet perlu membangun sistem agar dapat berkerja maksimal. Pada beberapa penelitian telah melakukan pengujian optimasi waktu *query* menggunakan *server* yang mereka bangun sendiri untuk suatu respon kinerja

dari *server* yang dibuat. Namun pembuatan *server* yang mereka bangun menggunakan *hardware* atau alat yang tergolong mahal dan susah dicari karena tidak dijual dipasaran dan tidak sembarang orang dapat memilikinya karena merupakan alat untuk suatu pengelolaan *environment* yang besar. Seperti contohnya alat *ML 30G9-069* paket *SMB Server hp Proliant* yang digunakan oleh penelitian sebelumnya [2]. Pada penelitian yang dilakukan kali ini menggunakan alat *Embedded System Raspberry Pi Model 3 B* sebagai suatu alternatif *Server* yang dapat menyelesaikan solusi yang sama dengan penelitian sebelumnya namun dengan harga yang relatif murah dan mudah didapatkan dipasaran untuk suatu *server* yang dibangun.

Server yang dibangun pada *Raspberry Pi* ini menawarkan penyelesaian *Resolver local -full service*, yaitu solusi yang dapat dijadikan suatu perbandingan dengan sistem kerja dari *Shared Resolver* yang mana menyediakan layanan lengkap lokal yang melibatkan pengalihan fungsi *Shared Resolver* dari *Domain Name System* di Internet dan dapat meningkatkan suatu kinerja jaringan yang lebih baik. Selain itu pada penelitian ini juga melakukan analisis tentang pengaruh *Resolver local -full service* dan *Shared Resolver* berdasarkan lalu lintas kinerja jaringan yang didapatkan dari *program automate traffic request.* Hal ini diharapkan dapat menawarkan kemungkinan kinerja jaringan yang lebih baik dengan *server* pada *Raspberry Pi.*

1.2. Topik dan Batasannya

Pada Penelitian ini akan dilakukan penelitian untuk melihat apakah penggunaan *Embedded System Raspberry Pi* sebagai *Resolver local-full service* pada suatu *server* jaringan dari sisi *client* dapat meningkatkan suatu kinerja jaringan yang dapat menyederhanakan sistem. Adapun *dataset* yang digunakan pada penelitian ini adalah *capture traffic* dari kesuluruhan keadaan jaringan menggunakan *Resolver local-full service* dan menggunakan *Shared Resolver*. Keduanya melakukan *capture traffic* dengan melakukan *program automate traffic request* dengan modul perintah *get request* menggunakan Algoritma *Python.*

Karena terlalu banyak topik yang dapat dibahas dalam penelitian ini, maka perlu membatasi hal pada penelitian topik ini dengan, hanya melakukan konfigurasi pada jaringan *server local*, hanya melakukan analisis dari segi kinerja jaringannya, proses *forwarding* dan *filtering* dari kedua tipe pengujian, tidak melakukan konfigurasi serta sinkronisasi terhadap *web server*, dan penelitian ini juga tidak melakukan pengujian serta analisis dari segi keamanan.

1.3. Tujuan

Tujuan dari penelitian ini adalah menganalisis serta membuktikan pengaruh keadaan suatu jaringan menggunakan Uji Kinerja jaringan, serta mencari tahu pengaruh permintaan DNS dengan Uji *Forwarding* dan Uji *Filtering* menggunakan *Resolver local-full service* dan *Shared Resolver*. Untuk mengetahui keoptimalan dari suatu *Resolver local-full service* yang dibangun maka perlu dilakukan analisis dan membandingkan hasil kinerja terhadap kedua *dataset* menggunakan *Resolver local-full service* dan yang menggunakan *Shared Resolver* dengan beberapa parameter pengujian antara lain *Quality of Service*, waktu DNS *Response Time* dan DNS *Retransmission* dari segi pengujian kinerja jaringan, serta melakukan pengujian *server* DNS dengan perintah *dig* pada suatu alamat *website* untuk uji *Forwarding* dan pengujian *blocking website* dari sisi *client* untuk uji *Filtering.* Diharapkan pada penelitian ini akan mendapatkan hasil keadaan kinerja jaringan yang terbaik agar dapat dimanfaatkan sebagai informasi dan solusi terbaik.

1.4. Tulisan

Pada Bab 2 akan menjelaskan Studi dan teori terkait dalam penelitian ini yang dapat mendukung proses pengerjaan ini. Selanjutnya Bab 3 akan memaparkan rancangan kegiatan untuk mendapatkan hasil dari pengerjaan ini. Bab 4 berisi hasil pengujian dan analisis untuk membuktikan keakuratan yang digunakan pada penelitian yang dilakukan. Serta Bab 5 akan menjelaskan kesimpulan dari hasil pengujian yang dilakukan pada penelitian ini disertai dengan saran untuk beberapa penelitian selanjutnya.

2. Studi Terkait

2.1. Related Work

Beberapa penelitian yang terkait dengan pengerjaan ini telah dilakukan di beberapa paper luar dari berbagai sumber. Namun terdapat beberapa perbedaan dari segi pengambilan data, olah hasil, serta sistem operasi yang digunakan sebagai *server* utama. Tetapi tetap tidak melupakan tujuan utama yaitu memperkuat lalu lintas DNS.

Penelitian Pertama ini memiliki tujuan utama yaitu menghapus *Shared DNS Resolver* dan menggantinya dengan Resolusi Rekursif. Pada penelitian pertama ini hanya menghilangkan suatu target untuk mengurangi kerentanan yang diketahui dan tidak diketahui. Penelitian pertama ini memerlukan perubahan pada protokol dan

menghasilkan ekosistem yang kurang kompleks secara keseluruhan dan tidak menambah komplesitas dengan mengurangi kerentanan *resolver* melalui modifikasi pada ekosistem DNS yang secara alami menambah kompleksitas pendekatan. Hasil dari penelitian ini adalah terdapat kurang dari 10% koneksi TCP tertunda oleh *resolusi* klien langsung. Selain itu terdapat beban yang tidak meningkat sama sekali untuk sekitar 90% di *server* ADNS dan dengan kedua faktor dari *server* TLD.com [1].

Pada penelitian kedua dengan judul "Optimasi Waktu Query Dan Filtering Nama Domain Pada Dns Server Lokal Menggunakan Bind 9", dibuat perancangan sebuah sistem jaringan yang dapat mengakomodasi kebutuhan informasi yang cukup besar dari pengguna. Penelitiaan yang dilakukan ini menggunakan alat *ML 30G9-069* paket *SMB Server hp Proliant* dengan sistem operasi *Debian 8 Squid 3, Iptables 1.414* sebagai server lokal mereka. Dari penelitian ini didapatkan hasil bahwa dengan diterapkannya DNS *forwarding* pada *router* yang sekaligus sebagai DNS *server* lokal mendapatkan hasil yang baik dengan hasil teknik *forwarding* di DNS *server* lokal 192.168.0.1, *query time* pertama mendapatkan nilai sebesar 274 *milisecond* dan untuk yang kedua sebesar 0 *milisecond*, sedangkan prngujian tanpa teknik *forwarding* mendapatkan *query time* pertama sebesar 53 *milisecond* dan pada pengujian kedau mendapatkan nilai sebesar 64 *milisecond*. Pada penelitian ini juga dilakukan 2 metode pengambilan data yaitu degan *forward* dan *filtering*, hasilnya pengujian melalui perintah *ping*, *nslookup*, *tracert*, dan melalui *browser* menghasilkan parameter Ya yang artinya kesemua nya berhasil dan berjalan dengan baik.[2].

Penelitian terakhir ini membahas tentang pengaruh dan dampak dari penghapusan *Shared Resolver* berdasarkan lalu lintas jaringan kampus dan menggantinya dengan *Resolver local full-service*. Analisis penelitian ini mengungkapkan bahwa terdapat efek *cache hit* dari *Shared DNS Resolver* berdasarkan analisis lalu lintas jaringan kampus yang diperkirakan bahwa terdapat pengaruh sistem *cache* lokal berdasarkan lalu lintas yang sama. Penghapusan bagian yang tidak perlu dari sistem secara keseluruhan dapat menyederhanakan suatu sistem. Dan memberikan kesimpulan analisis bahwa penggantian dengan *cache* lokal diharapkan dapat memperkuat lalu lintas DNS. Penghapusan semua sistem *cache* dari Internet kemungkinan besar akan memperkuat lalu lintas DNS sekitar 12,1 kali [3].

2.2. Cara Kerja DNS

Mengubah *hostname* menjadi sebuah alamat IP merupakan sebuah proses dari resolusi DNS. Sebuah alamat IP akan dibutuhkan agar dapat menemukan perangkat internet yang sesuai. Maka dari itu dilakukan pencarian DNS yang mana hal ini terjadi di belakang layer dan tidak memerlukan sebuah interaksi anatara komputer pengguna. Dalam pencarian DNS ini ada empat tahapan yang dilakukan untuk suatu proses pemuatan halaman website [4]

1. *DNS Recursor*
   Server ini bertanggung jawab atas segala bentuk pembuatan permintaan tambahan dan memenuhi kueri DNS klien. *Server* ini dibuat agar dapat menerima permintaan melalui *web browser* dari klien
2. *Root Name Server*
   Server ini berfungsi untuk acuan atau rujukan ke sebuah lokasi yang lebih spesifik. *Root Name Server* merupakan langkah awal untuk menerjemahkan suatu *host name* yang dibaca pengguna ke dalam pengalamatan IP.
3. *TLD Name Server*
   Server ini merupakan tahap selanjutnya setelah melalui *Root name* server yang merupakan bagian terakhir dalam suatu hostname dan berfungsi untuk mencari alamat sebuah IP.
4. *Authoritative Name Server (ADNS)*
   Authoritative Name Server adalah tahap akhir untuk proses permintaan *nameserver*. Jika dalam proses permintaan *nameserver* tersebut terdapat akses ke *record* yang diminta, maka alamat IP untuk *hostname* akan dikembalikan ke DNS *recursor*

2.3. Raspberry Pi Model 3 B

Gambar 1.1 Raspberry Pi Model 3B

Raspberry Pi adalah perangkat papan tunggal yang berukuran kotak kecil seperti *hardisk* eksternal yang awalnya dikembangkan di Inggris oleh *Raspberry Pi Foundation*. *Raspberry Pi* termasuk kategori perangkat yang sempurna walaupun harganya yang relatif murah tetapi cukup kuat sehingga dapat dengan mudah menggunakannya untuk dijadikan alternatif *server* berukuran kecil dan dapat melayani *client* untuk memperoleh keadaan jaringan yang lebih baik. Keuntungan menggunakan *Raspberry Pi* sebagai *server* ini yaitu *user* berada di dalam jaringan sendiri dan biayanya tergolong murah. Pada penelitian ini penulis menggunakan *Raspbian* standar yang merupakan sistem operasi resmi untuk semua tipe *Raspberry Pi* [5]. Agar suatu pengujian berjalan lancar, maka perlu dilakukan pengaturan IP *Static* pada *port ethernet*. Hal ini dilakukan agar memiliki alamat IP yang sudah ditetapkan, karena *desktop client* akan diarahkan ke sana sebagai DNS *server*. Adapun spesifikasi dari *Raspberry Pi Model 3B* ini yaitu;

- *Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz*
- *1GB LPDDR2 SDRAM*
- *2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE*
- *Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)*
- *4 USB 2.0 ports*
- *Micro SD port for loading your operating system and storing data*
- *5V/2.5A DC power input*
- *Power-over-Ethernet (PoE) support (requires separate PoE HAT)*

2.4. BIND9 pada *Resolver Local Full-Service*

BIND9 yang penulis gunakan pada penelitian kali ini merupakan suatu *software* berbasis *Linux* yang membantu untuk membangun suatu *server local* yang diharapkan dapat mempengaruhi suatu keadaan jaringan yang didapat. *BIND9* itu sendiri merupakan kependekan dari *Berkeley Internet Name Domain*s yang diperkenalkan oleh *Kevin Dunlap* untuk 4.3 *BSD Unix Berkeley*. *Resolver Local full-service* merupakan penyelesaian utama yang digunakan pada pengerjaan Tugas Akhir ini. Dimana pada pengerjaan ini melakukan konfigurasi komputer klien menggunakan IP dari *Resolver Local full-service* menggunakan *Raspberry Pi Model 3B* yang telah terintegrasi *BIND9* didalamnya sebagai *server*.

Untuk membuat *Raspberry Pi* sebagai *server* perlu melakukan penginstalan *BIND9* ke *Raspberry Pi*. Setelah proses penginstallan, *Raspberry Pi* perlu melakukan beberapa langkah konfigurasi pada *file file* yang terdapat pada direktori */etc/bind*. *File file* pada BIND yang perlu di konfigurasi antara lain *named.conf.options,* dan *named.conf.local* yang mana garis besarnya adalah untuk menentukan *local zone* agar client mendapatkan kinerja jaringan terbaik.

2.5. Parameter *Quality of Service* untuk Uji *Kinerja Jaringan*

A. Throughput

Untuk mengukur nilai throughput digunakan persamaan 2.1 sebagai berikut:

$$Throughput\ (bps) = \frac{\text{Jumlah Data Yang Dikirim (bit)}}{\text{Wakti Pengiriman data (seconds)}} \qquad (2\text{-}1)$$

Dimana *throughput* adalah banyaknya paket data yang dapat dikirim per-satuan waktu. Semakin kecil nilai *throughput*, maka semakin buruk jaringan yang di bangun [7]. Berikut merupakan standarisasi *Throughput* menurut THIPON [8].

Tabel 2.1 Standarisasi *Throughput versi* TIPHON

Kategori Throughput	Throughput
Sangat Bagus	>1.200 Kbps
Bagus	700 – 1.200 Kbps
Sedang	338 – 700 Kbps
Jelek	0 – 338 Kbps

(Sumber: TIPHON)

B. *Packet Loss*

Untuk mengukur nilai *Packet Loss* diguakan persamaan 2.2 sebagai berikut:

$$Packet\ Loss\ (\%) = \frac{Packet\ Data\ Yang\ Dikirim - Packet\ Data\ Yang\ Diterima}{\text{Paket Data Yang Dikirim}} \times 100\% \qquad (2\text{-}2)$$

Dimana *packet loss* merupakan kegagalan dalam Sebagian paket data ke tujuan jaringan. Semakin kecil *packet loss*, maka semakin baik jaringan yang dibangun [7]. Berikut merupakan standarisasi *Packet Loss* menurut THIPON [8].

Tabel 2.2 Standarisasi *Packet Loss* versi TIPHON

Kategori PacketLoss	Packet Loss
Sangat Bagus	0%
Bagus	3%
Sedang	15%
Jelek	25%

(Sumber: TIPHON)

C. *Delay*

Delay adalah waktu yang dibutuhkan dalam proses transmisi data dari saat data dikirim sampai diterima oleh *receiver*[8]. Untuk mengukur delay digunakan persamaan 2.3 sebagai berikut:

$$Delay = \text{Tr} - \text{Ts} \qquad (2\text{-}3)$$

Dimana Tr adalah waktu penerimaan paket dalam detik, dan Ts adalah waktu pengiriman paket dalam detik [7]. Berikut merupakan standarisasi *delay* menurut THIPON

Tabel 2.3 Standarisasi *Delay* versi TIPHON

Kategori Degredasi	Peak Jitter
Sangat Bagus	<150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Jelek	>450 ms

(Sumber: TIPHON)

D. *Jitter*

Jitter adalah perbedaan selang waktu kedatangan antar paket di terminal tujuan atau dapat disebut variasi *delay* [7]. Untuk mengukur *jitter* digunakan persamaan 2.3 sebagai berikut:

$$Jitter = \frac{Total\ variasi\ Delay}{\text{Total Paket yang diterima}} \qquad (2\text{-}4)$$

Dimana nilai Total variasi delay diperoleh dari penjumlahan = (delay2-delay1)+(delay3 - delay2)+ ... (delay n – delay (n-1)). Berikut merupakan standarisasi *Jitter* menurut THIPON [8].

Tabel 2.4 Standarisasi *Jitter* versi TIPHON

Kategori Degredasi	Peak Jitter
Sangat Bagus	0 ms
Bagus	0 s/d 75 ms
Sedang	75 s/d 125 ms
Jelek	125 s/d 225 ms

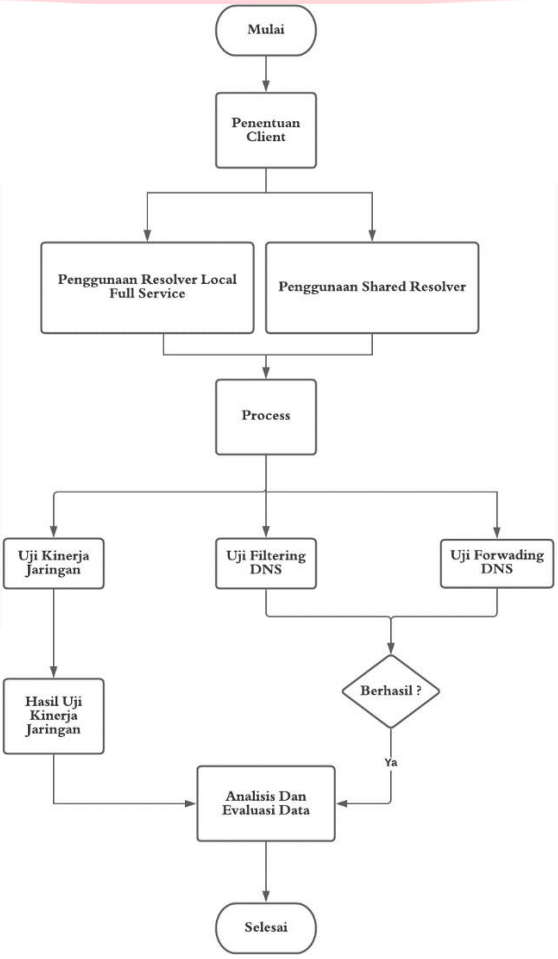
(Sumber: TIPHON)

2.6. DNS Retransmission

Retransmission merupakan suatu *respons*/permintaan yang disebabkan oleh hilangnya paket. Kurangnya konfirmasi dalam pengiriman protokol UDP membuat kebijakan *Retransmission* menjadi solusi kunci dalam keberhasilan aplikasi. Kebijakan ini tentu saja dapat dimodifikasi sesuai dengan kebutuhan efisiensi sebuah aplikasi. Kebijakan akan bervariasi sesuai kebutuhan dikarenakan lalu lintas internet yang tidak dapat diprediksi. Pendekatan yang optimal adalah dengan menentukan angka interval *Retransmission* berdasarkan data sebelumnya. Meskipun *Retransmission* mempunyai manfaat yang besar, tetapi ada konotasi negatif dalam implementasinya. Nilai *Retransmission* yang terlalu agresif dapat meningkatkan latensi bagi komunitas secara keseluruhan. Sebaliknya, nilai pasif berpotensi menyebabkan kehilangan paket yang berlebihan dengan tingkat pemulihan yang rendah. Interval Retransmit tipikal adalah antara 2 - 5 detik. Perilaku *Retransmission* UDP yang tidak menentu membuat pencarian keseimbangan netral menjadi sangat penting dalam implementasinya. Proses ini dapat dibantu dengan memanfaatkan kebijakan *Retransmission* UDP [9].

3. Sistem yang Dibangun

Pada tahap pembangunan sistem ini ada 3 tahap pengujian sistem yang dilakukan, yaitu pengujian kinerja jaringan, Uji *Forwarding* DNS, dan Uji *Filtering* DNS. Gambar 3.1 merupakan *flowchart* dari 3 tahap pengujian sistem yang dilakukan.

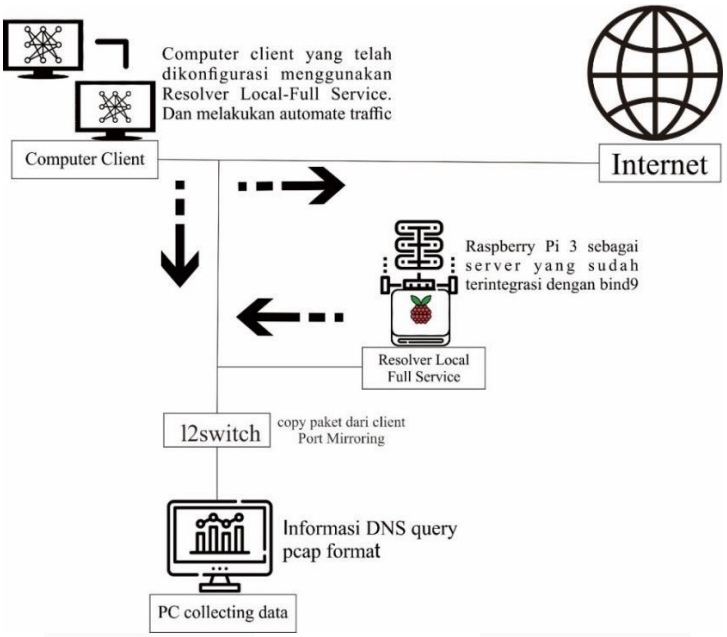


Gambar 3.1 *Flowchart* Skenario Pengujian

3.1. Alur Kerja Skenario Uji

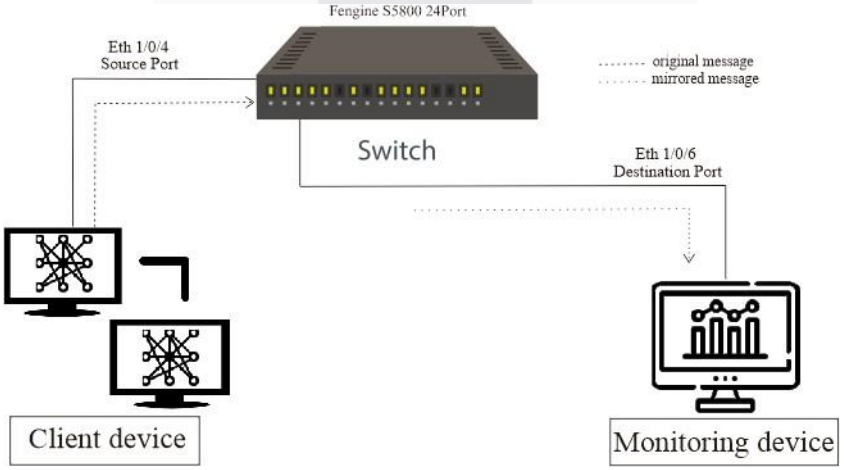
3.1.1. Uji *Kinerja Jaringan*

Untuk mengetahui keadaan jaringan yang didapat, perlu dilakukan uji kinerja jaringan dari suatu penerapan *Resolver local full service* dan *Shared resolver*. Pengujian kinerja jaringan ini dilakukan untuk mengetahui keoptimalan dari suatu *Embedded System Raspberry pi* sebagai *server* yang dibangun. *File* pengambilan paket yang di*capture* (pcap) merupakan generator utama untuk membuat suatu model pembuatan *query* untuk mengetahui kinerja jaringan. Sebelum mendapatkan *file pcap* tersebut perlu diperhatikan beberapa langkah yang harus dilakukan. Gambar 3.2 merupakan skenario pengumpulan data. Sebelum meng *capture traffic*, perlu dipastikan bahwa PC *client* dan PC *collecting* data sudah menggunakan IP *Raspberry Pi 3* yang sudah tergenarate sebagai *server* dengan BIND9 sebagai *Resolver Local Full Service.*



Gambar 3.2 Skenario Pengujian

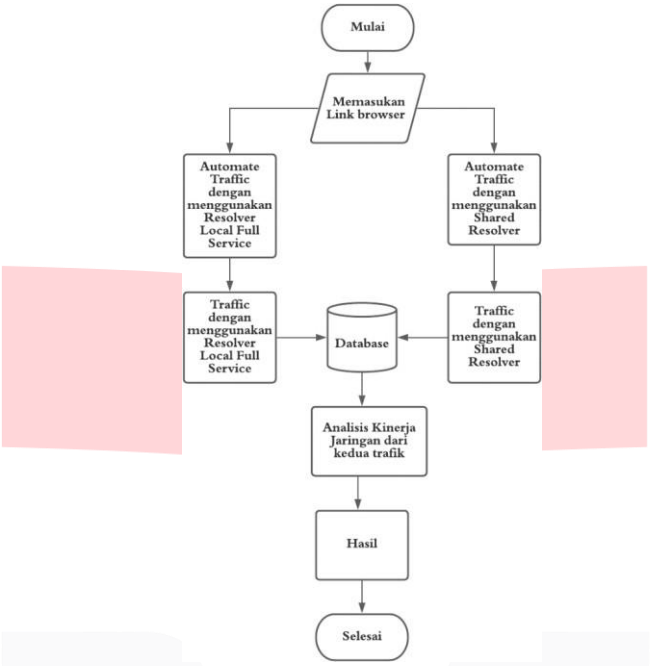
Harus dipastikan bahwa PC *client* perlu menggunakan jaringan yang sama dengan PC Pengumpulan data dan mengubah *Preferred DNS Server* dengan IP *Raspberry* pada *setting Network OS* yang digunakan. PC *Client* perlu menjalankan *program automate traffic request.* Disaat yang bersamaan PC Pengumpulan data menjalankan *Wireshark* dan meng*capture* nya.



Gambar 3.3 Topologi *Switch Port Mirroring*

Gambar 3.3 menunjukan bagaimana percakapan *client* dapat terbaca di PC *monitoring.* Perlu diingat dan dipastikan *client* menggunakan *port* berapa yang terhubung ke *switch*. PC *monitoring* melakukan konfigurasi *port mirroring* terlebih dahulu melalui *software Putty*.

Setelah *client* terdapat dalam *port mirroring*, Selanjutnya *client* melakukan akses *browsing* menggunkan *program automate traffic request* menggunakan *Resolver Local Full Service* dan *Shared Resolver*. Hasil kedua pengujian tersebut selanjutnya dianalisis hasil kinerja terbaik dari kedua pengujian. Gambar 3.4 menggambarkan bagaimana alur dari Uji Kinerja Jaringan bekerja.



Gambar 3.4 Flowchart Uji Kinerja Jaringan

A. Proses Pengumpulan Data Uji Kinerja Jaringan

1. Membuat *program automate traffic request.*
2. Mengumpulkan log lalu lintas dengan *automate traffic request* menggunakan Wireshark untuk setiap transaksi, mencatat setiap permintaan dan cap waktunya.
3. Membuat tautan koneksi dengan *query* DNS terdekat dari alamat IP dan jumlah lalu lintas pada suatu keadaan jaringan.

B. Parameter Pengujian Kinerja Jaringan pada permintaan DNS

Untuk mengetahui nilai kinerja jaringan terbaik maka perlu dilakukan empat proses pengujian. Keempat pengujian ini hasilnya berupa *file* pcap yang berisi *log traffic* yang mencatat setiap percakapan. Parameter yang menjadi suatu uji Kinerja Jaringan nya yaitu dari hasil *Throughput, Packet Loss, Delay, Jitter, Response Time* DNS dan DNS *Retransmission.* Adapun beberapa skenario pengujian kinerja jaringannya yaitu,

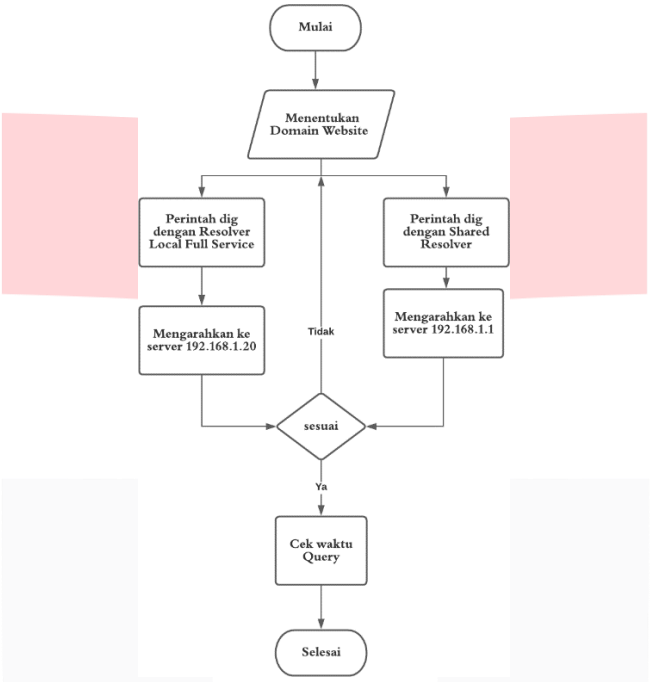
1. Menghitung *Response Time* DNS dan DNS *Retransmission.*
   - *a.* Dengan *Shared Resolver Cache* dan tanpa *Cache.*
   - *b.* Dengan *Resolver Local Full-Service Cache* dan tanpa *Cache.*

2. Menghitung *Throughput* dan *Packet Loss*
   - *a.* Dengan *Resolver Local Full-Service*
   - *b.* Dengan *Shared Resolver*

3. Menghitung *Delay* dan *Jitter* pada perintah *PING*
   - a. Dengan *Resolver Local Full-Service* (192.168.1.20)
   - b. Dengan *Shared Resolver* (192.168.1.1)
   - c. Dengan *Google Public* DNS (8.8.8.8)

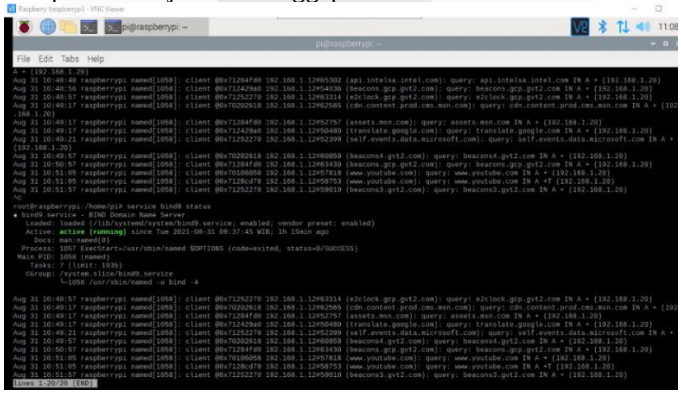
3.1.2. Uji *Forwarding DNS*

Forwarder merupakan fungsi dari pembuatan *server Resolver Local Full Service* pada jaringan yang bertugas untuk meneruskan permintaan DNS ke *server* yang berada di luar jaringan. Proses Uji *Forwarding* ditunjukkan oleh Gambar 3.5. Penggunaan *Bind9* dalam proses pengujian berfungsi sebagai *software* DNS *server* yang membantu untuk memungkinkan konfigurasi *forwarding* menggunakan *forwarder* pada basis per *zone* dari berkas *db.home.lan* dan *named.conf.local*. Untuk DNS *zone* itu sendiri merupakan bagian dari *database* DNS yang bertugas untuk memuat *resource record*. Pada uji coba ini terdapat beberapa langkah yang dilakukan untuk mengetahui kinerja dari konfigurasi yang telah dilakukan pada *server Resolver local full service* di *Raspberry Pi*, dengan menggunakan perintah *dig* untuk mengetahui *query time* dan menggunakan perintah *nslookup* untuk mengetahui *server* yang dituju yang bertujuan untuk menguji *query time* pada suatu DNS *server* lokal [2].



Gambar 3.5 Uji Forwarding

Pada pengujian ini terdapat parameter yang digunakan untuk mengukur kinerja *forwarding* adalah dengan melakukan pengujian menggunakan IP *Resolver Local Full-Service* 192.168.1.20 dan digunakan perintah *dig* pada sisi *server local* dan *query time* akses ke suatu alamat *website* sebanyak dua kali, lalu dibandingkan dengan perintah *dig* pada sisi *router* menggunakan IP DNS *Shared Resolver* yaitu 192.168.1.1. Jika uji coba kedua lebih cepat maka uji coba dianggap berhasil.

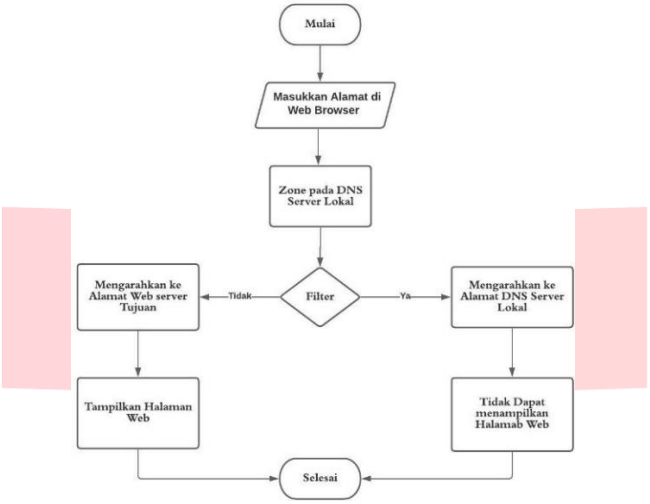


Gambar 3.6 Keadaan Jaringan pada Server Raspberry Pi

Gambar 3.6 menampilkan keadaan *log system* jaringan pada *raspberry pi* sebagai *Resolver Local Full Service* saat *server* menyelesaikan permintaan *query* pada suatu *resource record* yang diselesaikan dengan memberikan jawaban atas *client.*

3.1.3. Uji *Filtering* DNS

Untuk uji *filtering* melakukan percobaan akses internet dengan cara menambahkan *zone* pada berkas *named.conf.blocked* dan *db.blocked* yang alamatnya dialihkan pada sebuah *zone* yang mengarahkan alamat yang salah untuk *domain* yang dipilih yang dianggap melanggar kebijakan dari suatu *website* untuk proses uji *filtering*. Tampilan hasil dari *website* mengarahkan alamat *website* yang salah inilah nantinya akan ter*filter* dan tidak dapat menampilkan konten sesuai apa yang dikonfigurasi pada *server Resolver Local Full Service* pada *Raspberry Pi* yang akan mengarahkan permintaan klien pada alamat tertentu menuju alamat yang sudah diatur yaitu 192.168.1.20. Gambar 3.7 menjelaskan bagaimana uji *filtering* ini dapat bekerja.



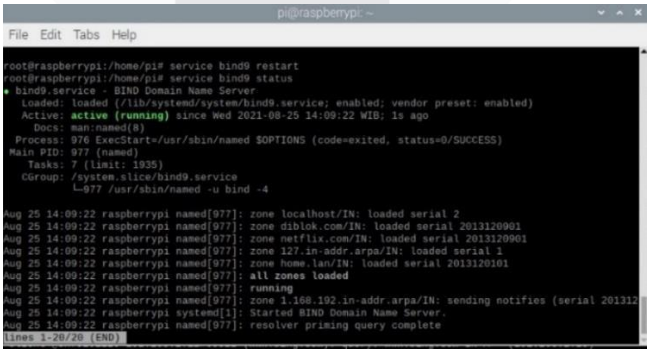
Gambar 3.7 *Flowchart* Uji *Filtering*

Pengunjian ini menggunakan beberapa perintah yaitu: *ping*, *nslookup*, *tracert*, dan akses melalui *browser* untu dilakukan pengujian terhadap DNS *server* lokal dan mengetahui kinerja dari konfigurasi yang dilakukan. *Browser* yang digunakan untuk pengujian ini adalah *www.netflix.com.* Terdapat beberapa parameter yang digunakan pada uji coba *filtering* ini untuk mengetahui apakah *filtering* berjalan dengan baik atau tidak.

Tabel 3.1. Parameter pengujian *filtering*

Perameter Uji			
Akses Browser	***Ping***	***Nslookup***	***Tracert***
Bisa / Tidak bisa	Bisa / Tidak bisa	Bisa / Tidak bisa	Bisa / Tidak bisa

Bisa : Jika koneksi terfilter dan dialihkan ke *server* 192.168.1.20.
Tidak Bisa : Jika koneksi gagal terfilter dan tetap diteruskan ke alamat asli.



Gambar 3.8 Keadaan Jaringan pada *Server* Uji *Filtering*

Gambar 3.8 merupakan tampilan keadaan *server* saat mengarahkan *zone* Netflix.com ke alamat *server Raspberry Pi* agar *client* tidak dapat mengaksesnya karena sudah ter block dari sisi *server*. Pengguna dapat mengakses penuh atas permintaan *client* untuk dapat terhubung langsung dengan *server*.

1. Evaluasi

4.1. Hasil dan Analisis Pengujian

A. Uji *Kinerja Jaringan*

Pengujian yang dilakukan dalam penelitian ini terdiri ke dalam 4 kategori pengujian kinerja jaringan. Hasil dari keempat kategori pengujian tersebut akan dibandingkan dan dianalisis untuk menemukan informasi serta kinerja jaringan terbaik. Selain *Throughput, Packet Loss, Delay, Jitter*, *Response Time* DNS dan DNS *Retransmission* ditampilkan juga banyaknya Jumlah keseluruhan paket dan Jumlah permintaan DNS pada sisi *client.* Tabel 4.1 dibawah ini menampilkan pengumpulan data dari keempat kategori pengujian kinerja jaringan. Terdapat waktu *capture* dalam satuan menit, jumlah keseluruhan paket, serta jumlah permintaan DNS (%). Baik buruknya kinerja jaringan yang didapatkan dari suatu penerapan *server* yang dibangun juga ditentukan dari suatu *Query Time Response* yang terjadi dari log lalulintas DNS. Parameter pengukuran tersebut juga ditentukan berdasarkan banyaknya jumlah permintaan DNS dari suatu kategori. Dari Tabel 4.1 juga dibawah ini menampilkan jumlah paket DNS *Response Time* dan DNS *Retransmission* yang didapatkan dari suatu *traffic* permintaan DNS berdasarkan keempat kategori yang dilakukan.

Tabel 4.1. Pengumpulan *Data Traffic*

Data Collecting	***Resolver local Full- Service***		***Shared Resolver***	
	Cache	**Tanpa *Cache***	***Cache***	**Tanpa *Cache***
Waktu *Capture* (Menit)	30	30	30	30
Jumlah Keseluruhan Packet	86.968	101.753	353.459	114.618
Jumlah Permintaan DNS (%)	1649 (1.9%)	1532 (1,5 %)	1586 (0.4%)	1384 (1.2%)
Rata rata DNS *Time Response*	56 ms	69 ms	31 ms	91 ms
DNS Retransmission	359 (0,4%)	413 (0,4%)	602 (0,2%)	960 (0,8%)

Dari Tabel 4.1 diatas didapatkan hasil yang berbeda beda untuk setiap permintaan dan paket yang didapat. Faktor jaringan dan *server* yang digunakan merupakan penyebab hasil dari keempat kategori ini menjadi beragam. Bisa dilihat dari Tabel 4.1 diatas bahwa dari segi permintaan DNS, *Resolver Local Full-Service* baik dengan *cache* ataupun tanpa *cache* mendapatkan hasil permintaan DNS yaitu 1,9 % dan 1,5 % dibandingkan dengan *Shared Resolver* yang hanya mendapatkan sekitar 0,4 % dengan *cache* dan 1,2 % tanpa *cache*. Dari hasil data ini bisa disimpulkan bahwa pembuatan *Resolver Local Full-Service* dapat mempengaruhi suatu kinerja jaringan pada permintaan DNS secara keseluruhan dari setiap kategori karna dilihat dari presentasi yang lebih besar dibandingkan dengan laju kinerja jaringan pada permintaan DNS menggunakan *Shared Resolver*.

Dari Tabel 4.1 diatas bisa dilihat juga bahwa dari segi parameter *Rata Rata Time Response* untuk *Resolver local Full- Service* baik dengan *cache* dan tanpa *cache* mendapatkan hasil kinerja jaringan dengan rata rata 56 ms dan 69 ms. Sedangkan dengan *Shared Resolver* memperoleh kinerja jaringan dengan rata rata *response time* 31 ms dengan *cache* dan 91 ms tanpa *cache*. Meski begitu nilai dari Rata Rata *Time Response* dari sisi penggunaan *Resolver local Full- Service* masih baik dengan waktu *response* tidak melebihi 70 ms.

Lain halnya dengan DNS *Retransmission, Resolver local Full- Service* dengan *cache* mendapatkan hasil persentasi yang jauh lebih rendah dengan 359 *query dns retransimission* atau sekitar 0,4 % dari total keseluruhan *query* dibandingkan dengan *Shared Resolver* dengan *cache* yaitu mencapai 602 *query dns retransmisson* atau sekitar 0,2 %. Sama halnya dengan *Resolver local Full- Service* dengan *cache, Resolver local Full- Service* tanpa *cache* pun mendapatkan presentasi lebih kecil dibandingkan *Shared Resolver* tanpa *cache*, yaitu sebanyak 0,4 % atau 413 *query* dibandingkan dengan *Shared Resolver* tanpa *Cache* yaitu 960 *query* atau sekitar 0,8 %. Dari hasil ini didapatkan bahwa ternyata dengan adanya pembuatan *Resolver local Full- Service* disisi *client* dapat membantu mengurangi adanya DNS *Retransmission* pada suatu kinerja jaringan yang didapat. Ini artinya bahwa dengan dibuatnya *Resolver local Full- Service* meminimalisir *respons*/permintaan yang dikirim ulang karena adanya resiko paket tidak terkirim atau kehilangan paket.

Untuk mengukur nilai kinerja jaringan dari keempat kategori yang dilakukan, dilakukan penghitungan *throughput* dan *packet loss* yang didapat dari kinerja jaringan pada kedua kategoti yang dilakukan. Tabel 4.2 dibawah ini merupakan hasil dari *throughput* dan *packet loss* yang didapat dari kedua kategori.

Tabel 4.2. Nilai Parameter *Throughput* dan *Packet Loss*

Nilai	*Resolver local Full-Service*	*Shared Resolver*
Throughput (Kb/s)	328 Kb/s	329 Kb/s
Packet Loss (%)	0,66 %	0,28 %

Nilai *Throughput* dan *Packet loss* yang didapat diatas merupakan nilai dari jumlah kesuluruhan paket pada kinerja jaringan yang didapat. Dari Tabel 4.2 diatas bisa dilihat bahwa nilai *throughput* dari *Resolver local Full- Service* mendapatkan hasil *throughput* 328 Kb/s lebih sedikit dibandingkan dengan *Shared Resolver* dengan sebanyak 329 Kb/s. Nilai *Throughput* disini hanya berbeda 0,1 Kb/s. Untuk Kinerja jaringan dari sisi *Throughput* yang didapat ini masih tergolong kurang baik dari sisi penggunaan *Resolver Local Full Service*. Dilihat dari definisi dari *Throughput* itu sendiri yaitu jumlah total kedatangan seluruh paket yang sukses diamati selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut hasil ini bisa saja didapatkan sesuai lamanya pengujian.

Dari segi uji kinerja jaringan *Packet Loss* pada *Resolver local Full- Service* mendapatkan hasil yang lebih besar yaitu 0,66 % dan dibandingkan dengan *Shared Resolver* yang hanya mendapatkan 0,48 %. Lain halnya dengan nilai *throughput, packet loss* yang didapat dari keempat kategori tersebut menyimpulkan bahwa dengan *Resolver local Full- Service* mendapatkan nilai persentasi lebih besar dibandingkan dengan *client* yang menggunakan *Shared Resolver*.

Untuk mengetahui nilai Jitter dan rata rata *Delay* dari suatu uji kinerja jaringan dilakukan perintah *ping* ke alamat *kominfo.go.id.* Adapun tipe pengujian yaitu *Resolver local Full- Service, Shared Resolver* dan *Google Public DNS* seperti pada Table 4.3. dan Tabel 4.4. Jumlah *Request* dan *Reply* masing masing untuk setiap kategori yaitu 50 dan tidak ada terjadi *Requested Time Out* dari semua *server*. Berikut merupakan hasil dari Pengujian *Ping* yang dilakukan.

Tabel 4.3. *Data Ping*

PING	*Resolver Local Full-Service*	*Shared Resolver*	*Google Public DNS*
Request	50	50	50
Reply	50	50	50

Tabel 4.4. Perintah *Ping*

Ping Request	*Resolver Local Full-Service*	*Shared Resolver*	*Google Public DNS*
Rata Rata Delay (ms)	507 ms	508 ms	498 ms
Jitter (ms)	0 ms	0 ms	0 ms
Max Response Time (ms)	7,2 ms	857 ms	15 ms
Min Response Time (ms)	3,3 ms	1,3 ms	4,4 ms
Rata Rata Response Time (ms)	4,4 ms	21 ms	5,5 ms

Dari Tabel 4.4 diatas bisa dilihat bahwa ternyata nilai dari rata rata *response time* pada perintah *ping* untuk ketiga kategori pengujian kinerja jaringan diungguli oleh penggunaan *Resolver Local Full Service* yaitu 4,4 ms dengan nilai *max response time* 7,2 ms dan *min response time* 3,3 ms. Hasil ini jauh beda dengan perintah *ping* pada *Shared Resolver* yang memperoleh nilai rata rata *response time* nya yaitu 21 ms dengan nilai *max response time* 857 ms dan *min response time* 1,3 ms. Namun berbeda dengan *response time* dari perintah *ping* pada *Google Public DNS*, *response time* nya lebih lama dibandingkan pengujian lainnya dengan mencapai rata rata *response time* yaitu 5,5 ms dengan nilai *max response time* 15 ms dan *min response time* 4,4 ms. Lain Hal nya pada pengujian untuk mencari nilai rata rata *Delay* pada perintah *Ping* pada *server Resolver Local Full Service* mendapatkan perolehan nilai *delay* 507 ms. Nilai ini lebih kecil dibandingkan *server* lainnya yaitu dengan *Shared Resolver* 508 ms dan untuk *Google Public DNS* 498 ms. Meskipun Nilai dari Rata Rata *Delay* pada *server Resolver Local Full-Service* lebih kecil dibandingkan *server* lainnya, namun hasil *delay* ini masuk pada kategori jelek menurut versi TIPHON karena Rata Rata *delay* melebihi 450 ms. Begitupun untuk hasil Rata Rata Delay dari *server Shared Resolver* dan *Google Public DNS.* Dari Hasil *response time* semua tipe pengujian pada perintah *ping*, server *Resolver Local Full Service* memperoleh nilai *response time* lebih kecil. Ini artinya pembuatan *Resolver Local Full Service* disini bisa dikatakan berhasil dengan dapat menjawab atas permintaan *client* lebih cepat dibandingkan *response time* dari penggunaan *server Google Public DNS* dan *Shared Resolver*.

B. Uji *Forwarding* DNS

Uji coba pengujian perintah *dig* dilakukan sebanyak dua kali untuk satu parameter pengujian. Berarti ada sebanyak empat kali pengujian perintah *dig* yang dilakukan karena ada dua parameter uji yaitu *client* yang menggunakan *Shared Resolver* yaitu IP *Server Router* dan *client* yang menggunakan *Resolver Local Full-Service* yaitu IP DNS *Server Local.* Jika waktu *query time* pada percobaan kedua lebih sedikit dibandingkan percobaan pertama, maka pengujian dianggap berhasil. Itu artinya *cache* telah tersimpan di DNS *Server Local.* Selain itu jika percobaan perintah dig menggunakan *Resolver Local Full-Service query time* nya lebih sedikit dibandingkan dengan *client* yang menggunakan *Shared Resolver*, maka tujuan utama untuk penggantian *Shared Resolver* terpenuhi dan dapat dipertimbangkan karena pengujian dianggap berhasil juga sesuai dugaan awal.

```
C:\Users\abusa>dig kominfo.go.id

; <<>> DiG 9.16.19 <<>> kominfo.go.id
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28057
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;kominfo.go.id.                 IN      A

;; ANSWER SECTION:
kominfo.go.id.          3600    IN      A       202.89.117.244

;; AUTHORITY SECTION:
kominfo.go.id.          3600    IN      NS      ns1.kominfo.go.id.
kominfo.go.id.          3600    IN      NS      ns2.kominfo.go.id.

;; ADDITIONAL SECTION:
ns1.kominfo.go.id.      3600    IN      A       180.250.112.15
ns2.kominfo.go.id.      3600    IN      A       202.89.117.3

;; Query time: 46 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Tue Aug 31 10:07:04 SE Asia Standard Time 2021
;; MSG SIZE  rcvd: 126
```

Gambar 4.1. *Query time* pertama dengan *Shared Resolver*

```
C:\Users\abusa>dig kominfo.go.id

; <<>> DiG 9.16.19 <<>> kominfo.go.id
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43255
;; flags: qr rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;kominfo.go.id.                 IN      A

;; ANSWER SECTION:
kominfo.go.id.          3592    IN      A       202.89.117.244

;; Query time: 15 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Tue Aug 31 10:07:32 SE Asia Standard Time 2021
;; MSG SIZE  rcvd: 58
```

Gambar 4.2. *Query time* kedua dengan *Shared Resolver*

Dari percobaan pertama dengan penggunaan dari *Shared Resolver* menghasilkan waktu kuery 46 ms sepperti pada Gambar 4.1. Selanjutnya, Hasil uji coba pertama ini akan dibandingan dengan perintah yang sama sekali lagi. Dalam pengujiannya, perintah *dig* pada *Shared Resolver* menggunakan IP *router* 192.168.1.1 dengan alamat *website www.kominfo.go.id* tetap digunakan pada pengunjian kedua. Pada

percobaan kedua dihasilkan *query time* sebesar 15 ms, seperti Gambar 4.2. Hal ini sudah sesuai dengan dugaan awal bahwa percobaan kedua nilai *query time* nya lebih sedikit daripada percobaan pertama, ini artinya *cache* nya telah tersimpan.

Selanjutnya dilakukan pengujian sebanyak dua kali uji *forwarding* DNS menggunakan *server Resolver Local Full Service* pada *Embedded System Raspberry Pi* yang akan diarahkan dengan menggunakan IP *server local* 192.168.1.20. Dihasil kan *query time* sebesar 15 ms pada Gambar 4.3, sedangkan pada pengujian kedua dihasilkan *query time* 0 ms dapat dilihat pada Gambar 4.4. Dengan mengarahkan *client* ke IP *server local*, dapat mengurangi waktu *query* pada percobaan pertama dan kedua dibandingkan dengan menggunakan IP *router*.

```
C:\Users\abusa>dig kominfo.go.id

; <<>> DiG 9.16.19 <<>> kominfo.go.id
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27935
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ebb695167f3e2a5714fa5d66612d9bcbd3bed9d6b5f59cab (good)
;; QUESTION SECTION:
;kominfo.go.id.                 IN      A

;; ANSWER SECTION:
kominfo.go.id.          3600    IN      A       202.89.117.244

;; Query time: 15 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Tue Aug 31 10:02:36 SE Asia Standard Time 2021
;; MSG SIZE  rcvd: 869
```

Gambar 4.3. *Query time* pertama dengan *Resolver Local Full-Service*

```
C:\Users\abusa>dig kominfo.go.id

; <<>> DiG 9.16.19 <<>> kominfo.go.id
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43019
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 5e4b7eef9a43c3f372099514612d9bd1d81d0ff7fab82051 (good)
;; QUESTION SECTION:
;kominfo.go.id.                 IN      A

;; ANSWER SECTION:
kominfo.go.id.          3594    IN      A       202.89.117.244

;; Query time: 0 msec
;; SERVER: 192.168.1.20#53(192.168.1.20)
;; WHEN: Tue Aug 31 10:02:42 SE Asia Standard Time 2021
;; MSG SIZE  rcvd: 869
```

Gambar 4.4. *Query time* kedua dengan *Resolver Local Full-Service*

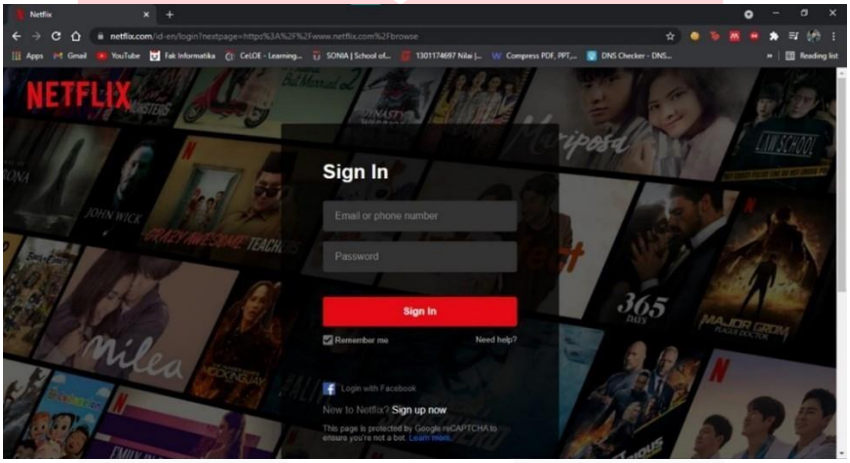
Penggunaan perintah ping dari sisi *client*, digunakan pada uji coba selanjutnya untuk melihat koneksi antara *client* dengan *web server* yang dituju. Pada Gambar 4.5 dapat dilihat bahwa hasil dari ping menunjukan perangkat dari pengguna dapat mengakses website yang dituju yaitu *kominfo.go.id* dengan alamat pada *server* nya yaitu 202.89.117.244.

```
C:\Users\abusa>ping kominfo.go.id

Pinging kominfo.go.id [202.89.117.244] with 32 bytes of data:
Reply from 202.89.117.244: bytes=32 time=6ms TTL=59
Reply from 202.89.117.244: bytes=32 time=5ms TTL=59
Reply from 202.89.117.244: bytes=32 time=5ms TTL=59
Reply from 202.89.117.244: bytes=32 time=5ms TTL=59

Ping statistics for 202.89.117.244:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 5ms, Maximum = 6ms, Average = 5ms
```

Gambar 4.5 Ping kominfo.go.id

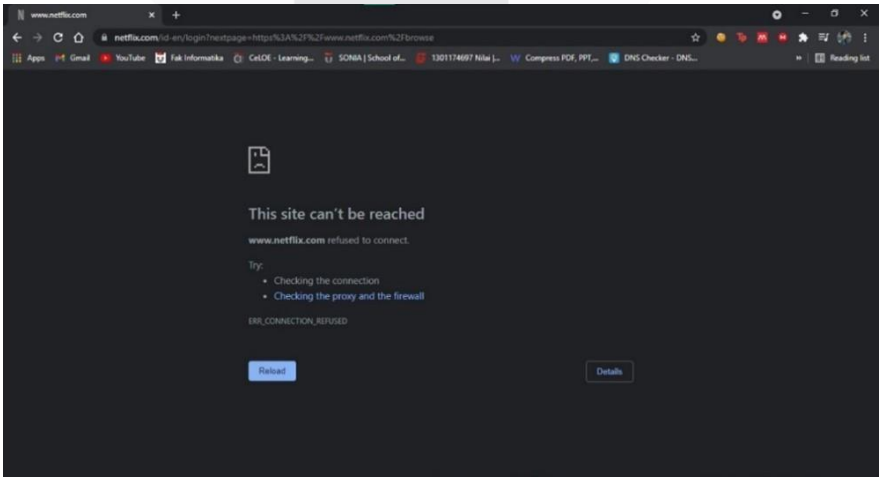
C. Uji *Filtering* DNS

Sebelum dilakukan *filtering* browser masih dapat diakses dan belum terfilter. Berikut adalah tampilan dari *website www.netflix.com* yang dapat dilihat pada Gambar 4.6.



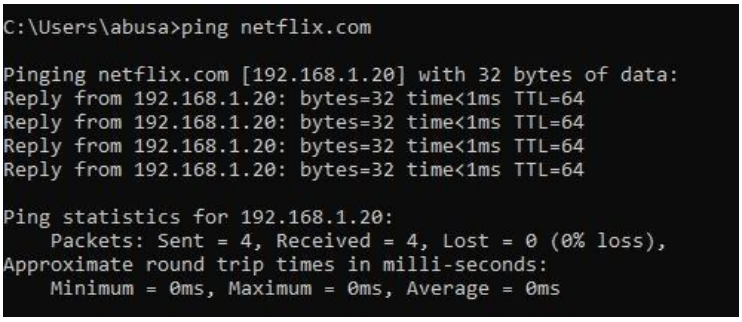
Gambar 4.6. Akses Browser Sebelum di Filter

Perlu dilakukan penambahan *zone host filter terlebih* dahulu pada *file named.conf.blocked.* Pada *case* ini penulis memasukkan penambahan *zone* alamat *netflix.com* untuk melakukan proses *filtering*. Gambar 4.7 menunjukan browser yang telah terfilter. Terlihat bahwa terdapat notifikasi *unable to connect*, konten yang ingin di akses oleh pengguna tidak dapat di tampilkan oleh *browser* yang berarti pengujian *filtering* berhasil dilakukan. Hal ini dapat diterapkan jika ada website yang kita anggap melanggar aturan dari sisi *client.* Untuk melakukan *unblock web browser* hanya perlu menghapus *zone host filter* pada server, maka browser dapat kembali diakses.



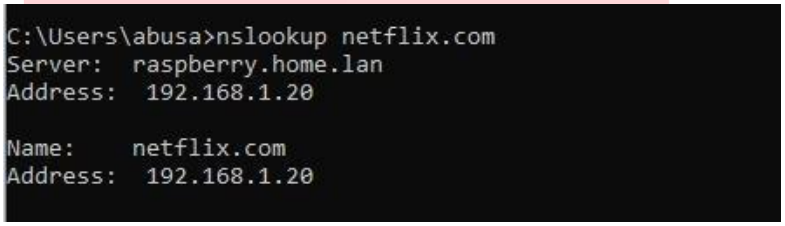
Gambar 4.7. Akses Browser sesudah di filter

Dilakukan juga pegujian perintah *ping* ke alamat *browser* yang ingin di *filter* untuk dapat diketahui komunikasi yang terjadi antara *server* tujuan dan *client*. Gambar 4.8 menunjukkan bahwa perintah *ping* menjawab *request* dan mengarahkan komunikasi ke alamat yang telah dikonfigurasikan yaitu 192.168.1.20.



Gambar 4.8. Ping *Filter* Netfilx.com

Selanjutnya dilakukan perintah menggunakan *nslookup*. Pada Gambar 4.9 menujukkan hasil yang didapat adalah alamat pada sisi *client* menggunakan alamat IP dari *Resolver Local Full Service*, sama halnya dengan *domain name* netflix.com mengarahkan alamat ke 192.168.1.20.



Gambar 4.9. Nslookup *Filter* Netfilx.com

Pada uji coba yang dilakukan terakhir yaitu menggunakan perintah *tracing route* atau *tracert*, seperti yang ada di Gambar 4.10. Hasilnya *client* hanya diarahkan pada 192.168.1.20 dan *tracing* selesai.

Gambar 4.10. Tracert filter Netflix.com

Berikut merupakan hasil akhir dari keempat pengujian yang dilakukan dari Uji *Filtering* Hasilnya semua pengujian bisa dilakukan dari keempat parameter.

Tabel 4.5. Hasil Uji *Filtering*

Perameter Uji			
Akses *Browser*	***Ping***	***Nslookup***	***Tracert***
Bisa	Bisa	Bisa	Bisa

2. Kesimpulan dan Saran

Kesimpulan yang didapat dari penelitian ini yaitu dari hasil semua tipe pengujian, bahwa dengan dibangunnya *Resolver local Full Service* pada suatu *Embedded System Raspberry Pi* memiliki beberapa keunggulan pada suatu hasil kinerja jaringan yang didapat dibandingkan dengan kinerja jaringan dari *Shared Resolver* di sisi *client.* Yaitu hasil pengujian kinerja jaringan pada jumlah permintaan dns dari sisi dengan *cache* dan tanpa *cache* sebanyak 1,9 % dan 1,5 %, dns *retransmission* dari sisi dengan *cache* dan tanpa *cache* sekitar 0,4 %, serta unggul dalam *time response* yang dihasilkan dari sisi tanpa *cache* 69 ms. Selain itu,

dibangunnya *Resolver local Full service* pada suatu *Embedded System Raspberry Pi* juga unggul dalam pengujian perintah *ping* pada nilai rata rata *response time* 4,4 ms, *max response time* 7,2 ms, dan *min response time* 3,3 ms. Dari sisi uji *forwading* juga tebukti bahwa dibangunnya *Resolver local full service* yang mengarahkan alamat perintah *dig* ke *server* 192.168.1.20 mendapatkan nilai *query time* pertama 15 ms dan kedua 0 ms, yang mana lebih kecil dibandingkan *Shared Resolver*. Dan dari sisi uji *filtering* dibangunnya *Resolver Local Full Service* berhasil mem *block website* secara mandiri dengan mengarahkan alamat yang salah. Dari hasil beberapa pengujian dapat disimpulkan bahwa dengan dibangunnya *Resolver Local Full Service* menggunakan *Raspberry Pi* menjadi suatu solusi yang baik bagi *client* dengan hasil *uji* kinerja jaringan yang tergolong baik serta dapat menghemat waktu pencarian *web server* dan menyimpannya ke *cache* atas jawaban dari *Resolver local full service* sebagai pembuatan *server* lokal. Dengan hasil ini *client* tidak perlu khawatir dengan suatu kinerja jaringan yang didapat atas penggunaan *server* lokal yang dibangun pada *Embedded System Raspberry Pi* dengan harga yang murah namun tetap dapat menghasilkan kinerja jaringan terbaik.

Saran yang dapat dilakukan untuk penelitian selanjutnya yaitu diharapkan pembuatan suatu *server* lokal ini dapat menghasilkan nilai yang terbaik hampir pada semua parameter kategori yang dilakukan dan lebih optimal jika dilakukan disuatu jaringan lokal pada *environment* yang besar dengan jumlah *client* yang banyak. Serta diharapkan dapat melakukan penambahan sistem keamanan yang dapat melindungi dari semua tipe penyerangan yang dapat membahayakan sistem.

REFERENSI

[1] R. ; Schomp Kyle , Allman Mark, "DNS Resolvers Considered Harmful," *Hotnets-XIII, Oct. 27–28, 2014, Los Angeles, CA, USA.*, 2014.

[2] D. Novianto, "Optimasi Waktu Query Dan Filtering Nama Domain Pada Dns Server Lokal Menggunakan Bind 9," *J. Ilm. Inform. Glob.*, vol. 8, no. 1, 2017, [Online]. Available: http://ejournal.uigm.ac.id/index.php/IG/article/view/320.

[3] K. Fujiwara, A. Sato, and K. Yoshida, "Cache Effect of Shared DNS Resolver," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 1, pp. 511–516, 2017, doi: 10.1109/COMPSAC.2017.77.

[4] Cloudflare, "What Is DNS? | How DNS Works | Cloudflare," *Cloudflare Inc.*, 2020. .

[5] J. F. Nusairat, *Rust for the IoT*. 2020.

[6] Cricket Liu and Paul Albitz, *DNS and BIND 5th Edition*. O'Reilly, 2003.

[7] F. T. P. W. Nindya Naraswari, Fitri Imansyah, "ANALISIS UJI KUAT SINYAL TERHADAP JARAK JANGKAU MAKSIMAL SISTEM PENERIMAAN SINYAL INTERNET BERBASIS EDIMAX HP-5101ACK."

[8] Y. Prabowo, H. Widiyantara, and P. Susanto, "Journal of Control and Network Systems," *JCONES J. Control Netw. Syst.*, vol. 3, no. 2, pp. 9–17, 2014.

[9] Ns1.com, "DNS Retransmission." https://ns1.com/resources/dns-retransmission (accessed Sep. 01, 2021).