

**PENINGKATAN EFISIENSI PENYIMPANAN DALAM
JARINGAN IOT MELALUI METODE *CACHE REPLACEMENT*
LRU (LEAST RECENTLY USED)**

SKRIPSI

GIDEON TULUS H Y SIAHAAN

201402116



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA**

2024

PENINGKATAN EFISIENSI PENYIMPANAN DALAM JARINGAN
IOT MELALUI METODE *CACHE REPLACEMENT* LRU (LEAST
RECENTLY USED)

SKRIPSI

Diajukan untuk melengkapi dan memenuhi syarat memperoleh ijazah
Sarjana Teknologi Informasi

GIDEON TULUS H Y SIAHAAN

201402116



PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

2024

PERSETUJUAN

Judul : PENINGKATAN EFISIENSI PENYIMPANAN
DALAM JARINGAN IOT MELALUI METODE
CACHE REPLACEMENT LRU (LEAST RECENTLY
USED)
Kategori : SKRIPSI
Nama : GIDEON TULUS H Y SIAHAAN
Nomor Induk Mahasiswa : 201402116
Program Studi : SARJANA (S-1) TEKNOLOGI INFORMASI
Fakultas : ILMU KOMPUTER DAN TEKNOLOGI
INFORMASI UNIVERSITAS SUMATERA UTARA

MEDAN, 04 Juli 2024

Komisi Pembimbing :

Pembimbing 2

Fahrurrozi Lubis B.IT., M.Sc.IT
198610122018052001

Pembimbing 1

Niskarto Zendrato S.Kom., M.Kom
198909192018051001

Diketahui/disetujui oleh :

Program Studi S-1 Teknologi Informasi

Ketua,

Dedy Arisandi S.T., M.Kom.
197908312009121002

PERNYATAAN

PENINGKATAN EFISIENSI PENYIMPANAN DALAM JARINGAN IOT MELALUI METODE *CACHE REPLACEMENT*

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing – masing telah disebutkan sumbernya.

Medan, 09 Januari 2024

Gideon Tulus H Y Siahaan
201402116

UCAPAN TERIMA KASIH

Penulis bersyukur dan berterimakasih kepada Yesus Kristus atas semua berkat dan pertolongannya penulis dapat melaksanakan penulisan skripsi dengan baik untuk memenuhi syarat agar lulus dan mendapatkan gelar Sarjana Komputer dari Program Studi Teknologi Informasi Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara. Penulis mengucapkan terimakasih kepada seluruh pihak yang berperan dalam penulisan skripsi ini hingga penulis mendapatkan gelar S1, tanpa semua pihak yang terlibat penulis tidak mampu menuntaskan penulisan skripsi ini. Adapun orang – orang yang terlibat, yaitu :

1. Bapak Dr. Muryanto Amin S.Sos., M.Si. sebagai Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. sebagai Dekan FASILKOM-TI Universitas Sumatera Utara.
3. Bapak Dedy Arisandy S.T., M.Kom. sebagai Ketua Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Sarah Purnamawati, ST, M.Sc. sebagai Dosen Pembimbing Akademik yang membantu saya selama saya menjalani perkuliahan di program studi S1 Teknologi Informasi Universitas Sumatera Utara.
5. Bapak Niskarto Zendrato S.Kom., M.Kom sebagai Dosen Pembimbing 1 dan Bapak Fahrurrozi Lubis B.IT., M.Sc.IT sebagai Dosen Pembimbing 2 yang telah memberikan arahan, saran dan motivasi kepada penulis.
6. Bapak Seniman S.Kom., M.Kom sebagai Dosen Penguji 1 dan Ibu Rossy Nurhasanah S.Kom., M.Kom sebagai Dosen Penguji 2 yang telah memberi saran, serta kritikan kepada penulis.
7. Keluarga penulis yaitu Bapak Mariyanto Siahaan, S.H. Ibu Eva Marthalina Sibarani, dan juga Kakak Gabriella Siahaan, S.H. dan juga keluarga besar penulis yang tidak dapat penulis tuliskan satu persatu yang senantiasa *mensupport* dan memotivasi penulis agar penulisan skripsi ini selesai.
8. Dosen Program Studi S1 Teknologi Informasi yang memberikan ilmu kepada penulis ketika masa perkuliahan.

9. Semua sivitas akademik Fakultas Ilmu Komputer dan Teknologi Informasi di Universitas Sumatera Utara yang membantu dalam segala urusan administrasi selama masa studi penulis.
10. Kepada BPH ITLG, yaitu Raihan Alifya Lubis, Karina Angela Tobing, dan Nadya Zahra yang telah membantu penulis selama diberi kepercayaan menjadi Badan Pengurus Harian.
11. Kepada teman – teman angkatan 2020 yang telah memberikan banyak pengalaman dan memberikan banyak bantuan kepada penulis sejak awal hingga akhir masa studi.
12. Teman – teman dekat yaitu Galileo Marapatama Purba, Ridha Arrahmi, Mhd. Daud Arbani Asfi Dalimunthe, Afifan Aly Rahman Saragih yang sudah membantu penulis selama masa perkuliahan sekaligus menjadi rekan di ITLG. Farhan Yehanda, Mustafa Kamal Nasution, Danny Panjaitan, Rheza Yudhistira Ramadhan yang sudah menjadi teman seperjuangan penulis dalam menulis skripsi.
13. Beberapa senior yang memberikan masukan, motivasi, ilmu dan bimbingan selama penulisan skripsi kepada penulis yaitu Bang Muhammad Saddam Zikri Dalimunthe, Kak Ineztri Situmeang, Kak Fenni Kristiani Sarumaha, Kak Sheren Alvionita Siahaan.
14. JKT48 sebagai grup idol asal Indonesia dan juga sebagai grup saudara AKB48 yang pertama di luar Jepang, yang telah memberi motivasi, inspirasi, semangat, dan dedikasi kepada penulis melalui lagu – lagu yang telah diciptakan sehingga penulis dapat menyelesaikan penulisan skripsi ini.
15. Gita JKT48 salah satu member JKT48 yang merupakan anggota favorit atau *oshi* penulis yang telah menjadi motivasi untuk penulis.
16. Semua yang terlibat dalam penulisan skripsi, yang tidak dapat penulis sebutkan satu per satu.

ABSTRAK

IoT (Internet of Things) merupakan jaringan perangkat yang saling terhubung dan dapat saling berkomunikasi. Karena perangkat tersebut terhubung langsung ke jaringan internet. Maka perangkat IoT memiliki sumber daya yang terbatas untuk mengakses data yang dimiliki. Data ini dapat disimpan dalam berbagai tempat, seperti di perangkat IoT itu sendiri, di server pusat, atau di cloud. Penyimpanan data di jaringan IoT sendiri merupakan tantangan tersendiri, karena data dapat berukuran besar dan dapat diakses dari lokasi manapun. Selain itu, jumlah pengguna yang semakin banyak juga mempengaruhi hal yang sama. Peningkatan ini berimbas pada kecepatan respons dari sistem dalam menangani *request* dari pengguna. Ketika pengguna ingin mengakses internet tentu saja membutuhkan perangkat pintar atau *gadget* yang terhubung ke web server. Dikarenakan penyimpanan yang terbatas baik itu penyimpanan data itu sendiri maupun penyimpanan untuk *cache* maka diperlukan suatu metode untuk mengganti data tersebut yang tersimpan terutama untuk penyimpanan *cache* itu sendiri. Maka dari itu, metode yang bisa digunakan untuk melakukan ataupun mengganti *cache* yang tersimpan yaitu dengan metode *cache replacement* dengan algoritma LRU (*Least Recently Used*). Adapun tujuan dari metode ini ialah untuk mempercepat akses dan mengoptimalkan penggunaan *cache* sehingga ketika web server ingin mengirimkan data yang diminta oleh pengguna maka *cache* tidak ikut terkirim dan *cache* tersebut sudah disimpan dan akan dilakukan metode *cache replacement* pada tempat penyimpanan *cache*. Pada penelitian yang dilakukan didapatkan hasil kesimpulan yaitu Metode Cache Replacement dapat diimplementasikan untuk keefektifitasan penyimpanan pada jaringan IoT dan juga algoritma LRU pada Metode Cache Replacement mampu mengurangi hit miss pada *cache*.

Kata kunci : IoT (Internet of Things), *Cache Replacement*, LRU (*Least Recently Used*), *Cache*

INCREASING STORAGE EFFICIENCY IN IOT NETWORKS THROUGH THE LRU (LAST RECENTLY USED) CACHE REPLACEMENT METHOD

ABSTRACT

The Internet of Things (IoT) is a network of devices that are interconnected and can communicate with each other. Because the devices are connected directly to the Internet, the IoT devices have limited resources to access the data they own. This data can be stored in a variety of locations, such as on the IoT device itself, on a central server, or in the cloud. Storing data on an IoT network itself is a challenge, as data can be large and accessible from any location. Besides, the growing number of users also affects the same thing. This enhancement scans the response speed of the system in dealing with the user's request. When a user wants to access the Internet, of course needs a smart device or gadget that is connected to a web server. Because of the limited storage of both the data storage itself and the storage for cache it is necessary a method to replace the stored data primarily for storing the cache itself. Therefore, the method that can be used to do or replace a stored cache is by cache replacement method with the LRU (Least Recently Used) algorithm. Whatever the purpose of this method is to speed up access and efficiently use cache so that when the web server wants to send the data requested by the user then cache will not be sent and cache has been stored and will be carried out cache substitution method at the location of cache storage. The research has concluded that the Cache Replacement Method can be implemented for the effectiveness of IoT and LRU algorithms in Cache replacement methods to reduce miss hits on cache.

Keywords: IoT (Internet of Things), Cache Replacement, LRU (Least Recently Used), Cache

DAFTAR ISI

PERSETUJUAN	ERROR! BOOKMARK NOT DEFINED.
PERNYATAAN.....	IV
UCAPAN TERIMA KASIH	V
ABSTRAK	VII
ABSTRACT	VIII
DAFTAR ISI.....	IX
DAFTAR TABEL	XI
DAFTAR GAMBAR.....	XII
BAB 1 PENDAHULUAN	1
1.1 LATAR BELAKANG	1
1.2 RUMUSAN MASALAH.....	2
1.3 TUJUAN PENELITIAN.....	3
1.4 BATASAN MASALAH	3
1.5 MANFAAT PENELITIAN.....	3
1.6 METODOLOGI PENELITIAN.....	3
1.7 SISTEMATIKA PENULISAN	4
BAB II LANDASAN TEORI	6
2.1 INTERNET OF THINGS.....	6
2.2 SENSOR DHT11	6
2.3 ESP8266	6
2.4 KABEL JUMPER.....	7
2.5 DATA STREAM	7
2.6 CACHE.....	7
2.7 CACHE REPLACEMENT	9
2.8 SERVER.....	10
2.8.1 <i>Web Server</i>	10
2.8.2 <i>Mail Server</i>	11

2.8.3 Database Server.....	11
2.8.4 File Server	11
2.8.5 Game Server	11
2.8.6 Application Server	12
2.8.7 Streaming Server.....	12
2.8.8 Proxy Server	13
2.8.9 DNS Server	13
2.9 PENELITIAN TERDAHULU.....	13
BAB 3 ANALISIS DAN PERANCANGAN.....	24
3.1 ANALISIS MASALAH.....	24
3.2 ANALISIS KEBUTUHAN SISTEM	24
3.3 ANALISIS DATA	25
3.4 ARSITEKTUR UMUM	25
3.4.1 Meminta Data	26
3.4.2 Mengirim Data.....	27
3.4.3 Mengambil Cache.....	28
3.4.4 Menyimpan Cache	29
BAB 4 IMPLEMENTASI DAN PENGUJIAN SISTEM.....	33
4.1. IMPLEMENTASI SISTEM	33
4.1.1. Software dan Hardware	33
4.1.2. Alat.....	33
4.1.3. Tahap Pengerjaan.....	34
4.2. PENGUJIAN UKURAN DATA.....	40
4.2.1 Ukuran Data Tanpa Ada Proxy Server	41
4.2.2 Ukuran Data Sebelum Menerapkan Algoritma LRU	42
4.2.3 Ukuran Data Setelah Menerapkan Algoritma LRU	43
BAB 5 KESIMPULAN DAN SARAN.....	45
5.1 KESIMPULAN	45
5.2 SARAN.....	45

DAFTAR TABEL

Tabel 2.1	Penelitian Terdahulu.....	16
Tabel 4.1	Permintaan data dan ukuran data hari pertama	41
Tabel 4.2	Permintaan data dan ukuran data hari kedua.....	41
Tabel 4.3	Perbandingan Efisiensi Cache dan Data Sebelum Menerapkan Metode Cache Replacement.....	42
Tabel 4.4	Perbandingan Efisiensi Cache dan Data Sesudah Menerapkan Metode Cache Replacement.....	43

DAFTAR GAMBAR

Gambar 3.2 Meminta Data.....	26
Gambar 3.3 Mini Server Mengirim Data	27
Gambar 3.4 Mengambil Cache	28
Gambar 3.5 Menyimpan Cache	29
Gambar 3.6 Flowchart Algoritma LRU	32
Gambar 4.1 Pin dht11 terhubung ke esp8266 kemudian dihubungkan ke laptop melalui usb cable	34
Gambar 4.2 Program untuk dht11	35
Gambar 4.3 Data suhu dan kelembapan pada thingspeak	36
Gambar 4.4 Penyimpanan cache dari data sensor.....	36
Gambar 4.5 Skrip Menyimpan Cache Thingspeak	37
Gambar 4.6 Skrip Untuk Melihat Ukuran Cache.....	38
Gambar 4.7 Skrip Untuk Algoritma LRU	39

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Kemajuan Teknologi Informasi mendorong peningkatan permintaan masyarakat akan komunikasi melalui jaringan internet. Berdasarkan survei APJII yang dilakukan pada Maret 2023, tercatat bahwa jumlah individu yang menggunakan internet di Indonesia telah mencapai 215.626.156, (APJII, 2023). Seiring bertambahnya jumlah pengguna internet, dibutuhkan bandwidth yang besar serta pengelolaan yang efektif untuk memastikan kelancaran kegiatan informasi dan komunikasi. Diperlukan sistem komunikasi yang lebih efisien karena mayoritas pengguna internet hanya tertarik pada konten yang mereka inginkan, tanpa mempedulikan sumbernya. Untuk memerlukan bandwidth yang cukup besar kita perlu memperhatikan beberapa hal seperti jenis jaringan, media transmisi, topologi jaringan, dan fungsi jaringan yang juga dapat mempengaruhi bandwidth jaringan atau internet. Jaringan sendiri dapat dikelompokkan menjadi beberapa jenis seperti LAN, MAN, WAN. Salah satu contoh untuk jenis jaringan LAN adalah jaringan sekolah, jenis jaringan MAN contohnya adalah jaringan di kota, dan untuk contoh jenis jaringan WAN ialah IoT. Jaringan WAN memiliki peran penting dalam aplikasi IoT, karena memungkinkan perangkat IoT untuk berkomunikasi berbagi data. (Al – Hamdani, et al. 2022)

(Pambudiatno, et al. 2021) mengatakan bahwa kemajuan teknologi saat ini berkembang dengan cepat baik perangkat keras dan perangkat lunak. Penulis juga mengatakan bahwa tujuan utama Internet of Things (IoT) adalah menghubungkan segala sesuatu yang ada di sekeliling kita, termasuk perangkat elektronik dan listrik, untuk memungkinkan komunikasi tak terbatas dan layanan kontekstual yang disediakan. Selain itu, perangkat ini biasanya terhubung melalui internet. Muafani (2020), mengatakan selama sepuluh tahun terakhir, komputer dan manusia hampir sepenuhnya bergantung pada internet untuk semua informasi. Melalui koneksi internet, kita dapat mengakses berbagai manfaat yang sebelumnya mungkin sulit diakses. Menurut Ayu dalam artikelnya di Cloud Computing Indonesia pada 17 Oktober 2020,

terdapat 31 miliar perangkat yang telah menerapkan sistem IoT dan saling terhubung satu sama lain hingga tahun 2020. Penerapan IoT sendiri dapat kita lihat pada berbagai kehidupan, seperti rumah pintar, kota cerdas, dan pintu yang terbuka otomatis.

Jaringan IoT dapat menghasilkan sejumlah besar data. Data ini dapat disimpan dalam berbagai tempat, seperti di perangkat IoT itu sendiri, di server pusat, atau di cloud. Penyimpanan data di jaringan IoT sendiri merupakan tantangan tersendiri, karena data dapat berukuran besar dan dapat diakses dari lokasi manapun. Selain itu, jumlah pengguna yang semakin banyak juga mempengaruhi hal yang sama. Peningkatan ini berimbas pada kecepatan respons dari sistem dalam menangani *request* dari pengguna. Ketika pengguna ingin mengakses internet tentu saja membutuhkan perangkat pintar atau *gadget* yang terhubung ke web server.

Penyimpanan pada web server tentu saja terbatas dikarenakan adanya data yang tersimpan, baik itu penyimpanan data ataupun penyimpanan *cache* (sampah) yang sudah tidak digunakan kembali. Banyaknya data yang perlu diakses ke dalam database adalah masalah umum sistem informasi berbasis web yang dapat mempengaruhi kinerja sistem (Jaelani, et al. 2023). Efisiensi penyimpanan web server penting karena dapat meningkatkan kinerja web server dan mengurangi biaya penyimpanan. Untuk itu diperlukan *cache replacement* yang bertujuan untuk mempercepat akses dan mengoptimalkan penggunaan *cache*. Metode *cache replacement* ini merupakan metode untuk mengganti data yang tidak lagi diperlukan dari *cache*. Metode *cache replacement* ini bertujuan untuk memilih objek yang terakhir diakses, objek dengan ukuran besar, dan objek yang sering diakses oleh *client* (Pratama, et al. 2020). Dengan mengganti data yang tidak digunakan lagi, *cache* dapat mempertahankan dalam ukuran yang efisien dan kinerjanya dapat ditingkatkan. Ada beberapa metode *cache replacement* yang dapat digunakan salah satunya yaitu metode LRU (*Least Recently Used*) yang dimana metode ini akan menghapus *cache* yang paling jarang digunakan. Untuk itu penulis melakukan penelitian dengan judul “Peningkatan Efisiensi Penyimpanan Dalam Jaringan IoT Melalui Metode *Cache Replacement* Dengan Algoritma LRU”

1.2 Rumusan Masalah

Dengan banyaknya pengguna internet saat ini, diperlukan peningkatan bandwidth agar kegiatan informasi dan komunikasi dapat berjalan lancar. Tantangan yang terjadi pada sistem informasi berbasis web adalah banyaknya data *cache* yang perlu diakses ke dalam database sehingga dapat mempengaruhi kinerja sistem. Selain itu jumlah

pengguna yang semakin banyak juga mempengaruhi hal yang sama. Peningkatan data *cache* ini berimbas pada kecepatan respons dari sistem dalam menangani *request* dari pengguna.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk memastikan bahwa data suhu dan kelembapan dapat diakses oleh semua orang, sehingga memungkinkan untuk memonitoring lingkungan secara real-time. Kemudian, untuk menerapkan algoritma LRU pada metode cache replacement guna meningkatkan efektivitas penyimpanan dalam jaringan IoT. Untuk mengurangi jumlah *cache miss* dengan menggunakan metode Cache Replacement dengan algoritma LRU, sehingga meningkatkan efisiensi dan kinerja sistem cache pada jaringan IoT. Untuk mengevaluasi efektivitas metode Cache Replacement LRU dalam penyimpanan jaringan IoT.

1.4 Batasan Masalah

Batasan masalah dari penelitian ini :

1. Waktu untuk melakukan *cache replacement* setiap 2 minggu.
2. *Cache* berada di mini server
3. Jarak terdekat dengan router adalah 50 m.
4. Limit cache sebesar 10 gb.

1.5 Manfaat Penelitian

Berikut manfaat dari penelitian ini :

1. Bagi pengguna, bisa merasakan efisiensi penggunaan *cache replacement* ini pada web server karena mampu menangani *request* dari pengguna dengan cepat.
2. Bagi penulis, bisa mengetahui bagaimana metode *cache replacement* ini bekerja.

1.6 Metodologi Penelitian

Langkah – langkah yang dijalankan dalam penelitian ini yaitu :

1. Studi Literatur

Penulis melakukan pengumpulan sumber referensi yang berasal dari jurnal, makalah, dan berbagai sumber yang berkaitan dengan penelitian penulis yaitu mengenai Metode *Cache Replacement* Dalam Penyimpanan Berbasis Jaringan IoT.

2. Analisis Permasalahan

Penulis melakukan analisis permasalahan yang telah dikumpulkan pada tahap sebelumnya yaitu dari berbagai sumber referensi.

3. Perancangan

Setelah melakukan analisis permasalahan penulis melakukan perancangan sistem yang bertujuan untuk membuat arsitektur umum sebagai perancangan program yang akan dibuat.

4. Implementasi

Perancangan sistem yang telah dibuat pada tahap sebelumnya diimplementasikan untuk menghasilkan metode yang sesuai berdasarkan tujuan penelitian.

5. Pengujian

Pada tahap ini uji coba dilakukan terhadap program dengan metode *cache replacement* yang telah dibuat untuk melihat hasil dari program berjalan sesuai dengan tujuan penelitian.

6. Penyusunan laporan

Penulis melakukan penyusunan laporan mengenai hasil dari penelitian yang dilakukan.

1.7 Sistematika Penulisan

Adapun sistematika penulisan pada penelitian ini terdiri atas lima bab, yaitu sebagai berikut :

Bab 1 : Pendahuluan

Bab pertama mencakup latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, metodologi penelitian, dan sistematika penulisan dari penelitian ini.

Bab 2 : Landasan Teori

Bab dua berisikan teori yang berhubungan dengan penelitian serta berkaitan dengan masalah dalam penelitian. Teori tersebut menjadi dasar pengetahuan yang diperlukan untuk memahami penelitian tersebut.

Bab 3 : Perancangan dan Analisis Sistem

Bab tiga berisi analisis data, perancangan arsitektur umum beserta penjelasan dari tahapan – tahapan yang terdapat didalam arsitektur umum penelitian serta melakukan perancangan pada metode yang akan digunakan.

Bab 4 : Implementasi dan Pengujian Sistem

Bab empat berisi pembahasan mengenai implementasi hasil dari perancangan yang telah dibahas pada bab sebelumnya. Dan juga bab ini terdapat hasil pengujian dari program yang telah dibuat.

Bab 5 : Kesimpulan dan Saran

Bab lima berisi rangkuman dari hasil penelitian yang sudah dilakukan dan juga saran untuk penelitian yang sejenis.

BAB II

LANDASAN TEORI

2.1 Internet of Things

IoT merupakan sebuah gagasan dalam bidang komputasi yang mengacu pada objek sehari – hari yang terkoneksi ke dalam jaringan internet dan bisa mengidentifikasikan diri ke perangkat lain (Muafani, 2020). IoT adalah cabang teknik yang dapat digunakan untuk menghubungkan objek fisik dengan dunia digital. Objek – objek tersebut dapat digunakan untuk mencapai tujuan yang lebih besar dan dapat digunakan untuk meningkatkan efisiensi dan produktivitas (Majid, et al 2022). Namun, menurut Villamil *et. al* (2020) *Internet of Things* (IoT) merupakan layanan yang dapat memenuhi berbagai kebutuhan pengguna, tergantung pada kemampuan perangkat yang digunakan. Elsaleh *et. al* (2019) menjelaskan bahwa perangkat (IoT) telah mengalami perkembangan yang sangat pesat sehingga menyebabkan terjadinya peningkatan aliran data yang diterima.

2.2 Sensor DHT11

Sensor yang dipergunakan untuk mengukur temperatur dan kelembapan adalah sensor dht11. Untuk menentukan suhu dan kelembapan, sensor dht11 terintegrasi dalam modul (Hadi, et al. 2022). Sensor tersebut mengeluarkan sinyal digital yang sudah dikalibrasi (Wardhani, et al. 2021). Yusuf *et.al* (2019) juga mengatakan bahwa sensor dht11 adalah sensor yang menggabungkan pengukur kelembapan dan suhu dalam satu chip kecil. Sensor ini sudah dikalibrasi dan menghasilkan data digital, sehingga mudah digunakan dan tidak memerlukan komponen tambahan seperti ADC. DHT-11 memiliki 14 bit untuk pengukuran suhu dan 12 bit untuk pengukuran kelembapan. Varian lain dari DHT-11 yang populer adalah DHT22 (AM2302).

2.3 ESP8266

ESP8266 merupakan chip Wi-Fi yang ekonomis yang memiliki kemampuan mikrokontroller dan tumpukan TCP/IP penuh. Komponen kecil ini memungkinkan mikrokontroller untuk terhubung ke jaringan Wi-Fi dan menjalankan koneksi TCP/IP sederhana menggunakan *AT command*. Harga yang sangat rendah dan jumlah komponen eksternal yang sangat sedikit pada modul menunjukkan bahwa komponen

dalamnya mungkin sangat murah untuk kita (Gunawan, et al. 2019). Santos *et.al* (2022) juga mengatakan bahwa esp8266 dipilih untuk menyelesaikan suatu penelitian karena memenuhi persyaratan untuk solusi yang murah, efisien, dan sederhana. NodeMcu ESP8266 dipilih sebagai controller utama karena merupakan controller *system-on-chip* (SOC) yang terintegrasi dengan Wi-Fi dan memiliki dukungan daya rendah, sehingga mengurangi biaya sistem yang disarankan. (Sutikno, et.al 2021).

2.4 Kabel Jumper

Kabel jumper ialah kabel yang menghubungkan mikrokontroller dan sensor yang akan digunakan. Bukan hanya menghubungkan antara mikrokontroller dengan sensor tapi bisa juga digunakan untuk menghubungkan breadboard dengan sensor. Menurut Yusuf *et.al* (2019) kabel jumper adalah kabel listrik yang bisa dipakai untuk menghubungkan komponen di breadboard tanpa menggunakan solder. Kalimat yang sama juga diungkapkan oleh Nabila *et. al* (2023) bahwa kabel jumper dapat menghubungkan komponen listrik pada papan breadboard tanpa menggunakan solder. Yusuf *et.al* (2019) dan Nabila *et.al* (2023) sama – sama menjelaskan bahwa kabel jumper memiliki *connector* atau pin di tiap ujungnya. *Connector* yang menusuk disebut *male connector*, sementara yang ditusuk disebut *female connector*. Untuk itu, kabel jumper dibagi tiga jenis, yaitu *male to male*, *male to female*, dan *female to female*.

2.5 Data Stream

Data Stream adalah aliran data yang terus menerus diproduksi oleh perangkat IoT, seperti sensor, kamera, dan perangkat cerdas lainnya (Elsaleh, et al. 2019). Kenda *et al.* (2019) juga mengatakan bahwa data streaming adalah data sensor dari perangkat IoT merupakan data yang terus mengalir dan dapat berasal dari berbagai sumber, baik perangkat, lokasi maupun waktu. Plaza *et. al* (2020) juga menjelaskan bahwa ada salah satu metode yang bisa digunakan untuk memproses data stream yaitu *stream processing*. *Stream processing* atau pemrosesan data stream adalah metode pemrosesan data yang dapat menangani data yang diterima secara terus – menerus, tidak seperti pada pemrosesan batch yang hanya dapat menangani data yang telah dikumpulkan sebelumnya. Pemrosesan data stream sangat penting untuk IoT karena data dari sensor dan perangkat lain terus – menerus bertambah.

2.6 Cache

Cache adalah ruang penyimpanan sementara yang dimanfaatkan untuk menyimpan data yang sering digunakan. Cache bisa ditemukan di berbagai sistem komputer, seperti pada CPU, memori, disk, dan jaringan. Cache juga bisa ditemukan pada proxy server. Cache pada proxy server adalah tempat penyimpanan objek yang dapat diakses secara cepat. Dengan adanya cache, objek yang diminta tidak perlu diakses dari internet secara langsung, tetapi dapat diakses dari cache terlebih dahulu (Tanwir, et. al 2021). Menurut Pratama et. al (2020) proxy server dapat meningkatkan performa akses internet dengan menyimpan objek – objek yang sering diminta oleh klien. Saat klien meminta objek yang sudah disimpan di proxy server, proxy server dapat langsung memberikan objek tersebut tanpa perlu mengakses internet lagi.

2.7 Cache Replacement

Metode cache replacement adalah teknik yang digunakan untuk mengelola cache dengan ruang penyimpanan terbatas. Ketika data baru perlu disimpan, namun cache sudah penuh, metode ini akan menentukan item mana yang akan dihapus untuk memberikan ruang bagi data baru tersebut. Metode ini banyak dilakukan oleh beberapa peneliti yang bertujuan untuk mengurangi cache yang akan terkirim ke client ketika client meminta data dari server. Seperti yang sudah dilakukan oleh Wijaya (2020) yang menggunakan algoritme cache proxy terdistribusi untuk meningkatkan performa server web dengan cara menggunakan cache pada server proxy web untuk meningkatkan cache hit dalam server proxy. Kemudian metode ini juga digunakan oleh Jaelani & Rosyid (2023) untuk melakukan implementasi application-level cache untuk optimasi kinerja web server dengan studi kasus SIM Surat PT. Gresik Migas. Pengujian pada penelitian ini dilakukan dengan membandingkan 2 metode cache, yaitu metode Object Level Cache, dan Page Level Cache. Berdasarkan kedua penelitian tersebut, dapat ditarik kesimpulan bahwa metode cache replacement dapat digunakan untuk meningkatkan kinerja dari web itu sendiri.

2.8 LRU (Least Recently Used)

Least Recently Used merupakan sebuah algoritma dalam Metode Cache Replacement yang dimana cara kerja dari algoritma ini adalah menghapus sebuah objek atau item yang sudah jarang digunakan atau tidak digunakan dalam waktu yang lama. Wang *et al.* (2019) menjelaskan bahwa algoritma LRU bekerja dengan cara menghapus objek yang paling jarang digunakan dihapus terlebih dahulu. Kemudian, peneliti juga mengatakan bahwa keuntungan dari LRU ialah mudah diimplementasikan. Kemudian,

Yovita *et al.* (2022) juga mengatakan bahwa algoritma LRU bekerja dengan cara menghapus konten yang sudah lama tidak diminta oleh pengguna sehingga diganti dengan konten yang baru. Peneliti juga mengatakan bahwa pada LRU konten yang dihapus adalah konten yang belum diakses untuk waktu yang lama. Nurjannah *et al.* (2019) menjelaskan bahwa LRU memantau paket interest yang akan dilayani beserta waktu pelayanannya. Ketika kapasitas penyimpanan *content store* mencapai batas maksimal, LRU akan menggantikan konten lama dengan yang baru. Konten yang paling lama tidak diminta oleh pengguna akan diprioritaskan untuk diganti. Suadi *et al.* (2020) menjelaskan bahwa algoritma LRU awalnya dirancang untuk pengelolaan cache dalam memori. Tujuan utamanya adalah untuk meningkatkan ratio hit dengan memanfaatkan lokalitas temporal. Peneliti juga mengatakan bahwa LRU unggul karena menyimpan data yang dianggap akan dipakai di masa mendatang dan juga LRU mampu menangkap aspek kekinian dari blok populer berdasarkan *temporal locality*. Tamwir *et al.* (2019) dalam penelitiannya juga menjelaskan bahwa Least Recently Used (LRU) adalah algoritma yang mengganti objek yang paling lama berada dalam cache dan tidak memiliki referensi.

2.8 Server

Server merupakan komputer yang bertugas melayani kebutuhan perangkat lain dalam jaringan, seperti menyediakan penyimpanan data, pemrosesan data, atau layanan lainnya. Menurut Michael *et.al* (2019) salah satu bagian utama sistem jaringan komputer adalah server, yang bertugas menyediakan layanan kepada client. Tujuan dari server yaitu menyimpan dan mengelola data, menjalankan aplikasi, menyediakan layanan. Server adalah media yang berperan penting dalam menyediakan layanan akses internet bagi pengguna. Untuk mengoptimalkan penggunaan jaringan, diperlukan adanya server. (Rizal, *et.al* 2022). Berdasarkan fungsinya, server dapat dibagi menjadi web server, mail server, database server, file server, game server, application server, streaming server, proxy server, dns server..

2.8.1 Web Server

Pada tahun 1989, Tim Berners-Lee menemukan web server, yang dia beri nama world wide web (www). Menurut Chandra (2019), web server adalah komponen penting dalam world wide web (www) yang memungkinkan website berjalan dengan baik dan memenuhi kebutuhan pengguna. Selain itu, perangkat lunak yang dikenal sebagai web server bertanggung jawab untuk menyimpan

dan menampilkan konten web pada pengguna. Server web merupakan sebuah aplikasi yang menyediakan layanan data dengan menerima permintaan dari klien dan mengirimkan informasi yang dibutuhkan dalam format halaman web serta kontennya. (Dwiyatno, et.al 2020).

2.8.2 Mail Server

Mail server merupakan server yang menyediakan layanan untuk mengirim dan menerima email. Menurut Sitorus et.al (2020) fungsi dari mail server sendiri dapat mengurangi biaya pengiriman surat – menyurat, lebih efektif, dan dapat menyertakan attachment yang berguna sebagai pelengkap dan dokumen tambahan terkait dengan isi email, sehingga mail server banyak digunakan di perusahaan.

2.8.3 Database Server

Basis data adalah kumpulan data yang terkait satu sama lain dan terorganisir secara sistematis guna memudahkan pencarian dan pengambilan data. Basis data dapat disimpan dalam komputer dan dimanipulasi dengan perangkat lunak basis data. Menurut Ramadhan et.al (2020) database adalah sebuah system yang dibuat untuk mengorganisasi, menyimpan dan menarik data dengan mudah. Fungsi dari database server adalah server yang menyediakan layanan untuk menyimpan dan mengelola data.

2.8.4 File Server

Fungsi dari file server yaitu server yang menyediakan layanan untuk menyimpan dan berbagi file. Untuk mengirim file antar dua komputer pada jaringan pada file server dapat menggunakan File Transfer Protocol (FTP). Menurut Supwanto et. al (2022) File Transfer Protocol (FTP) merupakan salah satu protokol internet yang dapat melakukan pertukaran data melalui sistem tersebut.

2.8.5 Game Server

Fungsi dari game server yaitu menyediakan layanan untuk bermain game online. Sadrif et.al (2023) mengatakan bahwa untuk mendapatkan pengalaman bermain yang optimal, memerlukan tantangan seperti menuntut performa yang tinggi dan skalabilitas infrastruktur yang baik. Untuk itu penulis memanfaatkan penggunaan cloud computing untuk meningkatkan performa, mengurangi latency, meningkatkan skalabilitas infrastruktur, dan mengurangi

biaya infrastruktur. Sementara itu Seo (2020) juga menjelaskan tentang desain server game online yang adaptif, *scalable*, dan *rebuildable* dengan komponen – komponen sehingga server tersebut terdiri dari 3 layer yaitu layer network yang bertanggung jawab untuk komunikasi dengan klien game, layer user yang bertanggung jawab untuk logika game dan interaksi dengan database, kemudian layer ketiga yaitu layer database bertanggung jawab untuk menyimpan data game.

2.8.6 *Application Server*

Fungsi dari application server yaitu server yang menyediakan layanan menjalankan aplikasi. Implementasi dari application server telah dilakukan oleh Costa et. al (2019) yang membahas sebuah algoritma adaptif untuk menyesuaikan thread pool pekerja secara otomatis di server aplikasi. Panduman, et. al (2023) juga menjelaskan implementasi dari application server dalam artikel penelitian yang membahas kerangka kerja perangkat edge (edge device framework) untuk platform server aplikasi IoT SEMAR (SEMAR IoT Application Server Platform) yang bertujuan untuk memudahkan pengembangan perangkat edge, meningkatkan skalabilitas dan fleksibilitas, serta meningkatkan keamanan dan keandalan.

2.8.7 *Streaming Server*

Fungsi dari streaming server yaitu menyediakan layanan untuk memutar video atau audio secara langsung Aisyah (2020) juga menjelaskan bahwa video streaming adalah cara untuk menonton video di internet tanpa harus mengunduh videonya terlebih dahulu. Kemudian video diubah menjadi format digital dan dipecah menjadi potongan-potongan kecil yang disebut "stream". Stream ini kemudian dikirimkan ke komputer atau perangkat lain secara real-time sehingga video tersebut dapat ditonton tanpa perlu menunggu sampai seluruh videonya diunduh. Untuk implementasi dari streaming server telah dilakukan oleh Hawari et al (2021), dalam jurnal yang diberi judul “Analisa Karakteristik Single Board Computer sebagai Streaming Video Server” penulis memberitahukan bahwa streaming video server Single Board Computer (SBC) dapat digunakan sebagai streaming video server dikarenakan Single Board Computer (SBC) yang berukuran kecil dan hemat daya, kinerja yang cukup untuk streaming video.

2.8.8 Proxy Server

Fungsi dari server ini yaitu server yang bertindak sebagai perantara klien dan server lain. Manfaat lain dari proxy server juga telah diteliti oleh Noviansya et.al (2020), penulis mengusulkan implementasi web proxy dengan router mikrotik untuk membatasi akses ke situs tertentu, seperti media sosial dan streaming, dengan tujuan meningkatkan disiplin dan efisiensi kerja.

2.8.9 DNS Server

Pederson et. al (2023) protokol penting jaringan komputer adalah Domain Name Server (DNS), yang mengubah nama domain menjadi alamat IP. Fungsi lain dari DNS Server yaitu server yang menyimpan dan merutekan informasi domain. Abdurahman et.al (2022) juga mengatakan bahwa dalam implementasinya DNS Server dapat dikombinasikan dengan PI HOLE sehingga penelitian tersebut berhasil dibuat dan menghasilkan PI Hole DNS Server yang bertujuan untuk membangun infrastruktur jaringan hotspot bebas iklan untuk mengoptimalkan penggunaan bandwidth, menerapkan sistem yang mencegah penyebaran virus melalui penyaringan situs web, serta meningkatkan kinerja router jaringan dalam jaringan hotspot. Dalam implementasinya, Pi Hole DNS server berfungsi pemblokir iklan dan sistem penyaringan situs web pada jaringan hotspot.

2.9 Penelitian Terdahulu

Penelitian sebelumnya dilaksanakan oleh Marvin Wijaya Chandra (2020), penelitian ini membahas tentang algoritme penggantian *proxy cache* yang terdistribusi digunakan untuk mengoptimalkan kinerja server web. Pada penelitian ini, peneliti menyimpulkan bahwa meningkatkan kinerja layanan dapat dicapai dengan menambah node ke node cache proxy web, termasuk dalam hal latensi dan throughput. Misalnya, dalam sistem kluster dengan 8 node, terdapat peningkatan latensi sebesar 90% lebih cepat dan throughput meningkat hingga 5,33 kali lipat. Penerapan kluster proxy web dapat meningkatkan kinerja layanan web dibandingkan dengan penggunaan proxy web tunggal.

Penelitian yang dilakukan oleh Arnold Nasir et. al (2023). Penelitian yang diberi judul Implementasi Proxy Server Untuk Optimalisasi Manajemen Bandwidth Jaringan Komputer Pada Universitas XYZ. Hasil dari penelitian menjelaskan bahwa penggunaan server proxy dalam manajemen bandwidth memiliki pengaruh besar terhadap pemakaian

jaringan di lingkungan kampus. Selain itu, hasil penelitian juga menunjukkan penggunaan proxy berkontribusi positif terhadap penyebaran yang lebih merata dalam kecepatan jaringan. Dengan menerapkan manajemen bandwidth yang efektif, lalu lintas jaringan dapat didistribusikan secara adil, sehingga penggunaan bandwidth tidak didominasi oleh sejumlah pengguna tertentu.

Penelitian lainnya juga dilakukan oleh Muhammad Nandaliyan Rais Al Azizi *et al.* (2022). Penelitian dilakukan dengan arsitektur NDN (Named Data Network) yang dimana arsitektur jaringan baru NDN berfokus pada data daripada model komunikasi yang berpusat pada host. Pengukuran CHR (Cache Hit Ratio) dihitung untuk mengetahui seberapa baik node pada jaringan NDN dapat menjawab paket interest yang diminta konsumen. Untuk mengetahui rata – rata waktu yang dibutuhkan oleh data atau konten selama transmisi antara node, RTT (Round Trip Time) diukur sebagai waktu yang dibutuhkan oleh data permintaan dari pengguna hingga mendapatkan tanggapan dari pembuat data. Kinerja terbaik ditunjukkan oleh nilai CHR yang tinggi dan RTT yang rendah. Tingginya nilai CHR menandakan bahwa node NDN berfungsi dengan baik karena mampu menyediakan konten yang diminta oleh pembuat tanpa perlu memintanya. Sedangkan, rendahnya nilai RTT menunjukkan bahwa NDN bekerja dengan efisien karena paket data dikirim dalam waktu yang lebih singkat.

Kemudian penelitian dilakukan oleh Tanwir *et al.* (2021) dengan judul penelitian Peningkatan Kinerja Jaringan Dengan Menggunakan Multi-Role Algorithm. Penelitian dilakukan dengan cara melakukan kombinasi algoritma pergantian cache FIFO-LRU-LFU (Multi-Rule Algorithm) pada ukuran cache 200 mb menghasilkan peningkatan hit rate sebesar 54,75% penurunan waktu penundaan sebesar 2 ms, peningkatan throughput sebesar 44,33 Kbps, dan waktu tanggapan sebesar 42 ms. Peningkatan hit rate ini berkontribusi pada percepatan permintaan dan pengurangan waktu respon. Kombinasi algoritma ini terbukti lebih efektif dibandingkan dengan algoritma FIFO, LRU, dan LFU yang digunakan secara terpisah atau dalam kombinasi lain, sehingga sangat direkomendasikan untuk meningkatkan throughput akses pada klien.

Leanna Vidya Yovita *et al.* (2022) melakukan penelitian dengan judul “Performance Analysis of Cache Replacement Algorithm using Virtual Named Data Network Nodes”. Penelitian ini menguji performa algoritma LRU dan FIFO dalam berbagai kondisi eksperimental menggunakan lima kali pengambilan data yang

berbeda. Pada pengambilan data pertama kondisi dengan berbagai ukuran penyimpanan konten, semakin besar ukuran penyimpanan konten menyebabkan peningkatan rasio cache hit (CHR). Perubahan ukuran ini juga berpengaruh terhadap waktu tunda pengiriman paket (RTT). LRU memberikan CHR yang lebih tinggi (5-12%) dan RTT yang lebih rendah (2-10%) dibandingkan FIFO. Pada pengambilan data kedua pengujian dengan variasi jumlah ketertarikan (permintaan dari pengguna) menunjukkan bahwa LRU memberikan CHR yang lebih tinggi (11-15%) daripada FIFO. Meskipun jumlah permintaan kecil masih dapat ditangani oleh penyimpanan konten ukuran 30, LRU lebih adaptif terhadap berbagai jumlah ketertarikan. Kemudian pada pengambilan data ketiga, peningkatan jumlah pengguna menyebabkan peningkatan CHR, dengan LRU memberikan peningkatan CHR yang lebih besar (6-12%) dibandingkan FIFO. Meskipun RTT pada LRU lebih rendah (1-3%) daripada FIFO, peningkatan jumlah konsumen tidak signifikan mempengaruhi RTT jika jumlah konten tetap sama. Data keempat diambil pada skenario dengan berbagai jumlah produsen, peningkatan jumlah produsen meningkatkan jumlah populasi konten dalam sistem. Meskipun ukuran penyimpanan konten 30 masih efektif hingga 5 produsen, peningkatan menjadi 10 produsen mengakibatkan penurunan CHR dan peningkatan RTT sekitar 3 ms untuk setiap dua kali lipat jumlah konsumen. Dan pengambilan data kelima dengan kondisi pada eksperimen dengan berbagai jumlah awalan (prefixes) untuk setiap produsen, hasilnya menunjukkan bahwa semakin tinggi jumlah awalan, semakin rendah CHR. Meskipun LRU tetap memberikan kinerja lebih baik dibandingkan FIFO (hingga 10% lebih tinggi), penurunan CHR disertai dengan peningkatan RTT.

Penelitian juga dilakukan oleh Elidio Tomás da Silva *et al.* (2022), penelitian ini membahas evaluasi kinerja kebijakan penggantian cache dalam Named Data Networking (NDN). Penelitian ini berfokus untuk mengevaluasi empat penggantian cache yaitu Least Recently Used (LRU), Least Frequently Used (LFU), dan First In First Out (FIFO), dan Random Replacement (RR) dalam berbagai skenario jaringan. Penelitian ini menggunakan beberapa metrik untuk mengevaluasi kinerja yaitu Cache Hit Ratio (CHR), lalu lintas jaringan, waktu pengambilan, retransmisi minat, dan jumlah hop yang digunakan. Permintaan konten mengikuti distribusi Zipf-Mandelbrot distribusi (dengan faktor kecondongan $\alpha = 1,1$ dan $\alpha = 0,75$). Hasil penelitian menunjukkan untuk tingkat penyimpanan konten dari 50% hingga 100% LRU

menunjukkan kinerja yang lebih tinggi. Meskipun perilaku jaringan serupa untuk faktor kecondongan ketika $\alpha = 0,75$ CHR berkurang secara signifikan.

Ringkasan penelitian terdahulu dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1 Penelitian Terdahulu

No.	Penulis	Tahun	Judul	Keterangan
1.	Marvin Chandra Wijaya	2020	Algoritme penggantian cache proxy terdistribusi untuk meningkatkan kinerja web server	Penelitian ini bertujuan untuk membahas penggunaan algoritme penggantian cache terdistribusi pada proxy web guna meningkatkan performa server situs web. Temuan dari penelitian menunjukkan bahwa penambahan node dalam cache proxy web bisa memperbaiki kinerja, seperti latensi dan throughput. Dalam contoh sistem kluster dengan 8 node, terdapat peningkatan kinerja, dimana latensi menjadi 90% lebih cepat dan throughput meningkat 5,33 kali lipat dibandingkan dengan menggunakan sistem web proxy tunggal. Hal ini

				membuktikan implementasi kluster proxy web secara efektif dapat meningkatkan kinerja layanan web.
2.	Arnold Nasir, Reinaldo Lewis Lordianto	2023	Implementasi Proxy Server Untuk Optimalisasi Manajemen Bandwith Jaringan Komputer Pada Universitas XYZ	Hasil penelitian menyimpulkan penggunaan server proxy dalam mengelola <i>bandwith</i> menimbulkan efek besar terhadap penggunaan jaringan di lingkungan kampus. Temuan tersebut juga menegaskan bahwa server proxy dapat membantu menjaga keseimbangan kecepatan jaringan secara positif. Dengan menerapkan manajemen <i>bandwith</i> yang efektif, penggunaan lalu lintas jaringan dapat tersebar secara merata, menghindari dominansi penggunaan <i>bandwith</i> oleh

				sejumlah pengguna tertentu.
3.	Muhammad Nandaliyan Rais Al Azizi, <i>et al.</i>	2022	Analisis Pengaruh Jumlah Konten Dan Node Dengan Cache Replacement LRU Pada Virtual Node NDN	Untuk penelitian ini digunakan arsitektur NDN (Nalmed Data Network). Arsitektur jaringan ini inovatif dan memiliki kemampuan untuk mengubah mode komunikasi yang berpusat pada data. Untuk mengevaluasi kinerja, Cache Hit Ratio (CHR) dan Round Trip Time (RTT) digunakan. CHR menunjukkan seberapa efektif, node merespons permintaan data, RTT menunjukkan waktu yang dibutuhkan dari pengiriman permintaan hingga tanggapan. Performa terbaik ditandai dengan CHR tinggi dan RTT rendah, yang menunjukkan bahwa node NDN dapat memberikan konten yang diminta tanpa

				meminta produsen data untuk melakukan permintaan.
4.	Tanwir <i>et. al</i>	2021	Peningkatan Kinerja Jaringan Dengan Menggunakan Multi-Role Algorithm	<p>Penggunaan kombinasi algoritma pergantian cache FIFO-LRU-LFU (Multi-Rule Algorithm) pada ukuran cache 200 mb menghasilkan peningkatan hit rate sebesar 54,75% penurunan waktu penundaan sebesar 2 ms, peningkatan throughput sebesar 44,33 Kbps, dan waktu tanggapan sebesar 42 ms. Peningkatan hit rate ini berkontribusi pada percepatan permintaan dan pengurangan waktu respon. Kombinasi algoritma ini terbukti lebih efektif dibandingkan dengan algoritma FIFO, LRU, dan LFU yang digunakan secara terpisah atau dalam</p>

				<p>kombinasi lain, sehingga sangat direkomendasikan untuk meningkatkan throughput akses pada klien.</p>
5	Yovita <i>et al.</i>	2022	Performance Analysis of Cache Replacement Algorithm using Virtual Named Data Network Nodes	<p>Penelitian ini menggambarkan sistem NDN yang lebih realistis dan mengevaluasi kinerjanya dengan menggunakan dua algoritma penggantian cache dalam berbagai skenario. Ukuran cache store (CS) mempengaruhi secara langsung rasio cache hit (CHR), dimana semakin besar CS, semakin tinggi CHR-nya dan RTT cenderung lebih rendah. Algoritma LRU menunjukkan superioritasnya dibandingkan FIFO dalam semua kondisi, dengan perbedaan CHR dan RTT yang semakin besar seiring dengan peningkatan</p>

				<p>ukuran CS. Jumlah permintaan dari pengguna juga memengaruhi CHR, meskipun dalam kasus tertentu dengan jumlah permintaan tertinggi, CHR dapat mengalami penurunan karena keterbatasan CS ukuran 30 dalam menampung variasi permintaan konsumen.</p> <p>Peningkatan CHR berkorelasi dengan penurunan RTT, walaupun tidak signifikan jika jumlah populasi konten tetap. Perubahan jumlah produsen menyebabkan CS harus menyimpan lebih banyak konten untuk mengakomodasi permintaan yang lebih bervariasi dari konsumen. Semakin tinggi jumlah awalan (prefixes), semakin rendah CHR-nya</p>
--	--	--	--	--

				<p>karena kesulitan CS dalam mengelola permintaan yang bervariasi. Hasil uji coba menunjukkan bahwa LRU secara konsisten memberikan kinerja lebih baik dalam parameter CHR dan RTT dibandingkan FIFO. Untuk pengembangan lebih lanjut, penelitian akan fokus pada implementasi sistem NDN menggunakan komputer tunggal untuk setiap node, mempersiapkan sistem untuk implementasi praktis di masa mendatang.</p>
6.	Elídio Tomás da Silva <i>et al.</i>	2022	NDN Content Store and Caching Policies: Performance Evaluation	<p>Penelitian ini berfokus untuk mengevaluasi empat penggantian cache yaitu Least Recently Used (LRU), Least Frequently Used (LFU), dan First In First Out (FIFO), dan Random Replacement (RR) dalam berbagai skenario jaringan.</p>

				<p>Penelitian ini menggunakan beberapa metrik untuk mengevaluasi kinerja yaitu Cache Hit Ratio (CHR), lalu lintas jaringan, waktu pengambilan, retransmisi minat, dan jumlah hop yang digunakan.</p> <p>Permintaan konten mengikuti distribusi Zipf-Mandelbrot distribusi (dengan faktor kecondongan $\alpha = 1,1$ dan $\alpha = 0,75$). Hasil penelitian menunjukkan untuk tingkat penyimpanan konten dari 50% hingga 100% LRU menunjukkan kinerja yang lebih tinggi. Meskipun perilaku jaringan serupa untuk faktor kecondongan ketika $\alpha = 0,75$ CHR berkurang secara signifikan.</p>
--	--	--	--	---

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis Masalah

IoT merupakan sebuah konsep dalam komputasi yang menghubungkan benda sehari-hari ke jaringan internet, memungkinkan mereka untuk mengenali dan berkomunikasi dengan perangkat lain (Muafani, 2020). Jaringan IoT dapat menghasilkan sejumlah besar data. Data ini dapat disimpan dalam berbagai tempat, seperti di perangkat IoT itu sendiri, di server pusat, atau di cloud. Penyimpanan data di jaringan IoT sendiri merupakan tantangan tersendiri, karena data dapat berukuran besar dan dapat diakses dari lokasi manapun. Semakin banyak data yang tersimpan semakin banyak cache yang tersimpan di perangkat IoT, server, ataupun pada penyimpanan cloud. Jika hal itu terjadi maka akan mengakibatkan volume data yang dihasilkan oleh perangkat IoT terus meningkat, yang mengakibatkan kapasitas penyimpanan data pada perangkat IoT tidak bisa mengikuti pertambahan atau peningkatan volume data. Dampak dari permasalahan tersebut yaitu performa jaringan IoT menurun, biaya penyimpanan data meningkat, dan juga membuat kekuatan jaringan IoT menurun. Untuk itu diperlukan solusi dari permasalahan tersebut, dan solusi yang dapat digunakan ialah meningkatkan efisiensi penyimpanan dalam jaringan iot dengan suatu metode dengan algoritma, yaitu metode cache replacement dengan algoritma LRU (Least Recently Used). Oleh karena itu, metode cache replacement dengan algoritma LRU (Lest Recently Used) diharapkan dapat menjadi solusi

3.2 Analisis Kebutuhan Sistem

Adapun tujuan dari metode cache replacement dengan algoritma LRU (Least Recently Used) yaitu meningkatkan efisiensi penyimpanan dalam jaringan IoT.

Dalam membangun sistem ini terdapat 2 kebutuhan yaitu kebutuhan fungsional, dan kebutuhan non-fungsional.

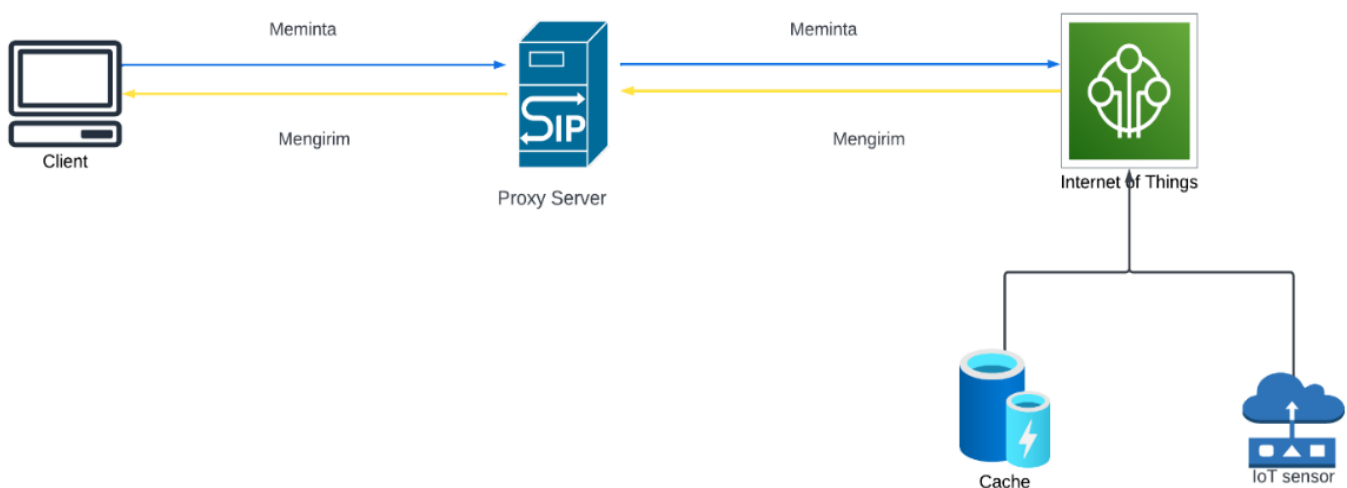
- a. Kebutuhan fungsional yaitu,
 - Sistem harus dapat menyimpan data sensor dari perangkat IoT.
 - Sistem harus dapat mengakses data sensor dengan cepat dan efisien.
 - Sistem harus dapat menerapkan metode cache replacement LRU.
- b. Kebutuhan non-fungsional yaitu,
 - Kinerja sistem harus dapat menangani permintaan data dengan cepat.
 - Sistem harus dirancang dengan cara yang memungkinkan untuk ditambahkannya lebih banyak perangkat IoT tanpa menurunkan kinerja sistem.

3.3 Analisis Data

Data yang digunakan dalam penelitian ini merupakan data sensor dari perangkat IoT dan juga data log sistem. Kemudian data yang didapatkan akan dilakukan analisis data dengan metode analisis statistik dan juga visualisasi data. Tujuan dari analisis data ini yaitu untuk memahami pola penggunaan data, mengidentifikasi data yang jarang digunakan, dan juga untuk mengevaluasi efektivitas metode cache replacemet LRU.

3.4 Arsitektur Umum

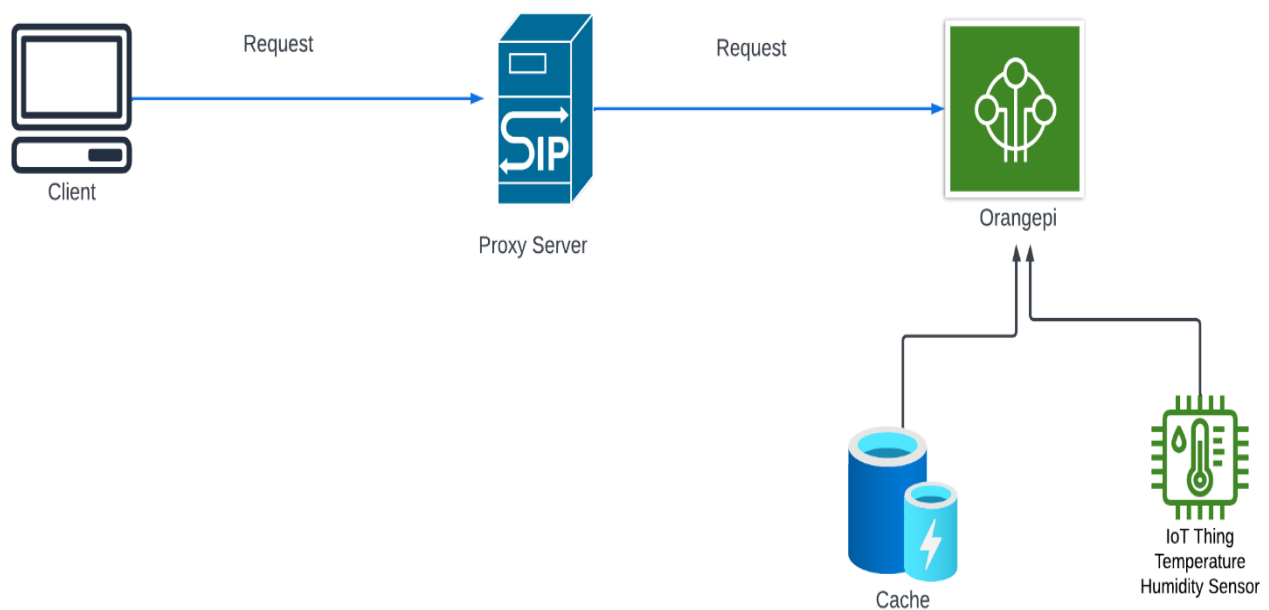
Adapun arsitektur umum dari penelitian sebagai berikut.



Gambar 3.1 Arsitektur Umum

Berikut ini penjelasan dari gambar 3.1

3.4.1 Meminta Data



Gambar 3.2 Meminta Data

Berikut penjelasan proses yang terjadi pada gambar saat client meminta data.

a. Client

Pada tahap ini client akan meminta request kepada mini server melalui proxy server dan data yang diminta oleh client merupakan data suhu dan kelembapan yang didapat dari sensor IoT suhu dan kelembapan.

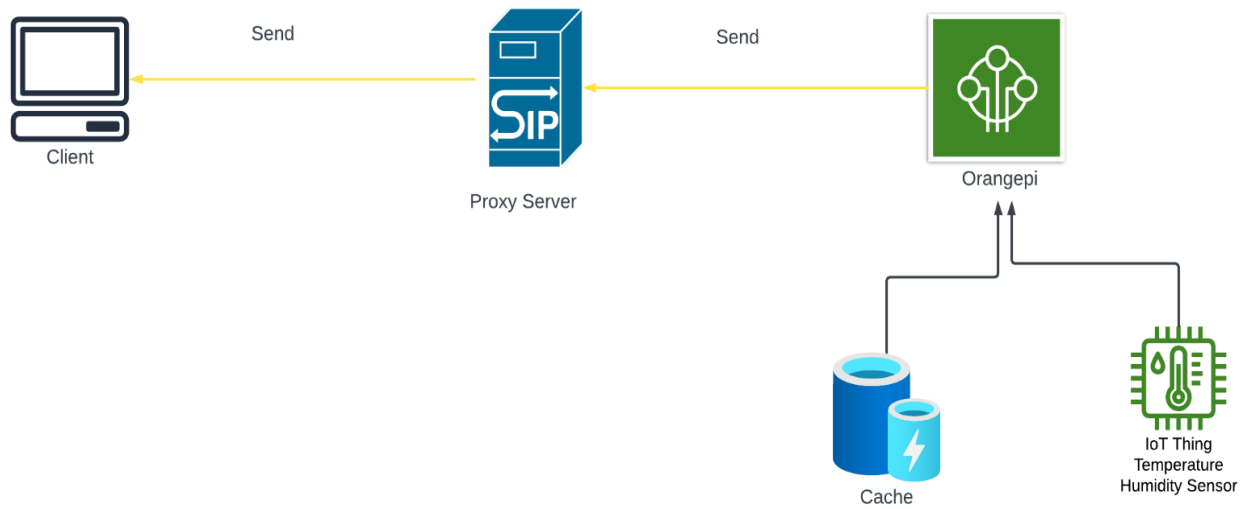
b. Proxy Server

Fungsi dari server ini yaitu server yang bertindak sebagai perantara klien dan server lain, yang berarti proxy server akan meminta data suhu dan kelembapan kepada mini server yang telah terlebih dahulu disimpan di dalam mini server.

c. Raspberrypi

Mini server yang digunakan pada penelitian ini yaitu raspberrypi yang berfungsi untuk menyimpan data sekaligus cache yang dihasilkan dari IoT sensor suhu dan kelembapan. Kemudian orangepi akan menerima request dari client melalui proxy server.

3.4.2 Mengirim Data



Gambar 3.3 Mini Server Mengirim Data

Berikut penjelasan proses yang terjadi pada gambar saat orangepi mengirim data kepada client.

a. Raspberrypi

Raspberrypi akan mengirim data kepada proxy server, data yang dikirim berupa suhu dan kelembapan suatu ruangan ataupun tempat. Namun sebelum data dikirim, cache akan disimpan terlebih dahulu agar data yang dikirimkan tidak ikut beserta cache tersebut. Di dalam cache tersebut akan dilakukan metode cache replacement agar penyimpanan cache tetap optimal.

b. Proxy Server

Proxy server akan mengirimkan informasi kepada client yang diperoleh dari sensor iot, dan data yang dikirimkan kepada client sudah tidak beserta cache karena cache pada data tersebut sudah dipisah dan disimpan dalam penyimpanan cache.

c. *Client*

Client akan menerima data suhu dan kelembapan dari sensor iot, tanpa ada cache di dalam tersebut, dikarenakan cache pada data tersebut telah disimpan ataupun telah dipisah sebelum dikirim kepada *client*.

3.4.3 Mengambil Cache



Gambar 3.4 Mengambil Cache

Raspberrypi dapat mengambil cache dengan beberapa cara:

1. Akses langsung

Jika cache disimpan dalam RAM, raspberrypi dapat langsung mengakses tanpa memerlukan operasi tambahan. Cara ini, merupakan cara tercepat untuk mengambil cache, namun data cache akan hilang ketika raspberrypi dimatikan.

2. Membaca dari penyimpanan

Jika cache disimpan dalam penyimpanan permanen seperti eMMC, RAM, maka raspberrypi perlu membacanya dari penyimpanan sebelum dapat digunakan. Cara ini lebih lambat untuk mengambil cache dibandingkan dengan akses secara langsung, namun cache akan disimpan secara permanen.

3. Mekanisme caching internal

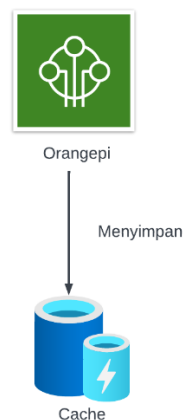
Ketika cache dibutuhkan mekanisme caching akan mencari di cache terlebih dahulu. Jika data tidak ditemukan, mekanisme cache akan mengambilnya

dari penyimpanan dan menyimpannya dalam cache untuk akses selanjutnya yang lebih cepat.

Berikut proses pengambilan cache :

1. Permintaan data : CPU atau aplikasi akan meminta data.
2. Pemeriksaan cache : mekanisme caching internal memeriksa cache untuk melihat apakah data tersedia.
3. Cache hit : jika data ditemukan dalam cache, data tersebut dikembalikan ke CPU atau aplikasi.
4. Cache miss : jika data tidak ditemukan dalam cache, mekanisme caching akan mengambilnya dari penyimpanan permanen.
5. Penyimpanan cache : data yang diambil dari penyimpanan permanen disimpan dalam cache untuk akses selanjutnya yang lebih cepat.
6. Pengembalian data : data dikembalikan ke CPU atau aplikasi.

3.4.4 Menyimpan Cache



Gambar 3.5 Menyimpan Cache

Raspberrypi dapat menyimpan cache dengan beberapa cara tergantung pada tipe cache seperti cache statis atau cache dinamis. Jika cache tersebut merupakan cache statis maka cache akan disimpan dalam file permanen dan tidak berubah frequently. Namun, jika cache tersebut merupakan cache dinamis maka cache dapat dibuat dan diubah secara real-time berdasarkan permintaan pengguna. Kemudian, cache juga bekerja pada lokasi penyimpanan internal yaitu RAM, eMMC, SPI flash. Dan juga bekerja pada lokasi penyimpanan eksternal yaitu kartu SD, USB drive, penyimpanan cloud. Adapun cara kerja dari raspberrypi untuk menyimpan cache yaitu raspbeeypi menyimpan data

yang sering diakses seperti file sistem, pustaka, dan data dalam aplikasi, dalam cache RAM. Kemudian, saat data (cache) dibutuhkan raspberry pi terlebih dahulu mencari cache di RAM, jika cache tidak ditemukan, raspberry pi akan mengambil dari penyimpanan permanen (eMMC, kartu SD) dan menyimpannya di cache RAM untuk akses selanjutnya yang lebih cepat.

3.4.5 Melakukan Replacement Cache

Ketika cache penuh dan data baru perlu disimpan, maka diperlukan metode cache replacement yang bertujuan untuk mengganti cache yang lama dengan cache yang baru agar penyimpanan cache tetap efisien. Mekanisme cache replacement akan menentukan data mana yang harus dihapus dari cache untuk memberi ruang baru bagi data baru. Untuk menggunakan metode atau mekanisme cache replacement algoritma yang peneliti gunakan ialah algoritma Least Recently Used (LRU). Algoritma ini bekerja dengan cara menghapus data yang paling lama tidak digunakan dari cache. LRU juga algoritma yang mudah untuk diterapkan dan cukup efektif dalam banyak permasalahan. Algoritma LRU digunakan untuk menggantikan elemen cache yang paling lama tidak digunakan ketika cache mencapai kapasitas maksimalnya. Algoritma ini menggunakan link list ganda untuk melacak urutan pengguna cache, serta hash map untuk akses cepat ke elemen cache berdasarkan kuncinya. Berikut merupakan pseudocode dari algoritma LRU.

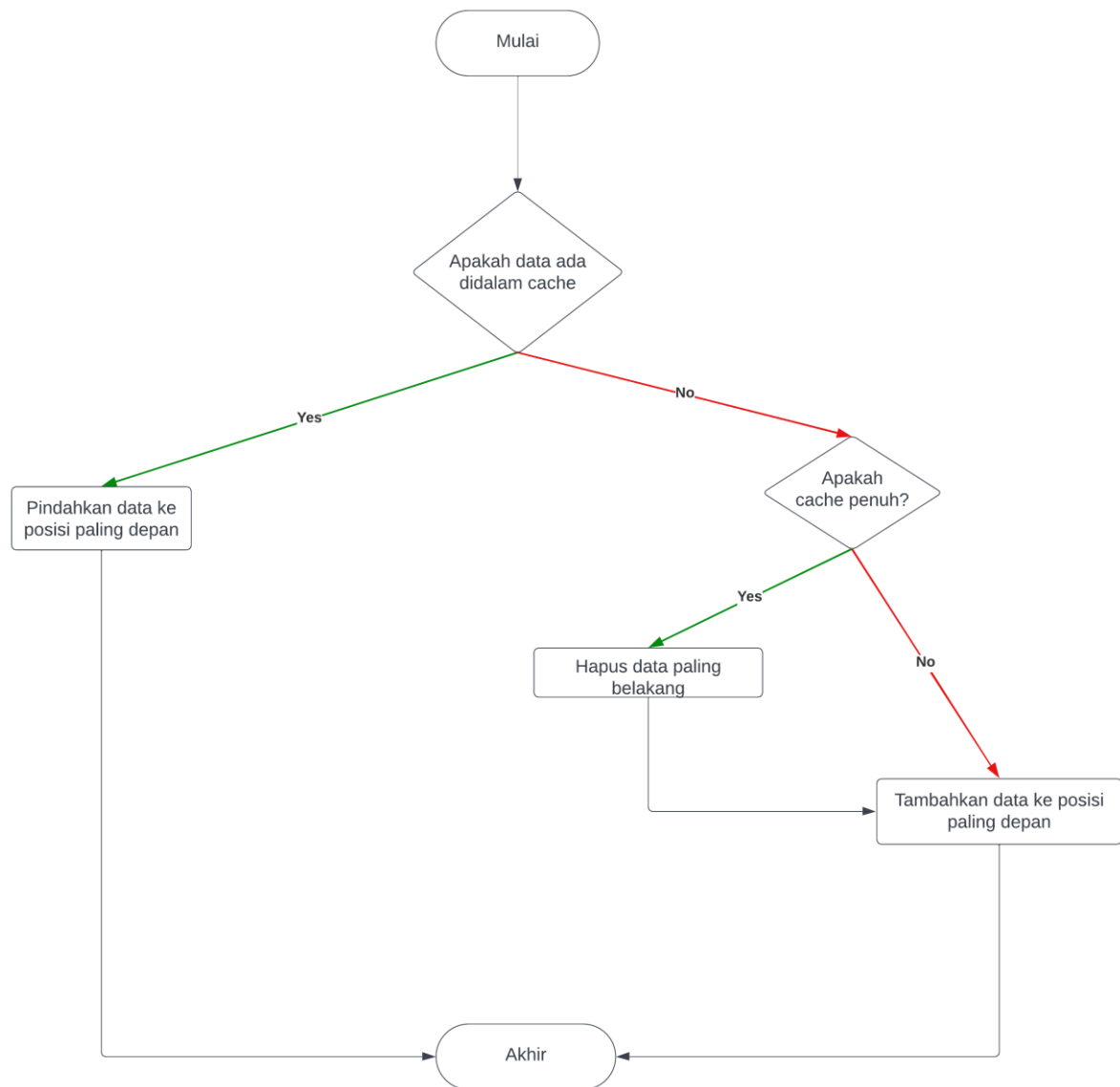
```

INITIALIZE cache as an empty data structure with a fixed size
INITIALIZE a doubly linked list to keep track of usage order
INITIALIZE a hash map to store key-node pairs for quick access
FUNCTION get(key):
    IF key is in cache:
        node = hash_map[key]
        move node to the front of the linked list (most recently used)
        return node.value
    ELSE:
        return -1 (or other indication that the key is not in cache)
FUNCTION put(key, value):
    IF key is in cache:

```

```
node = hash_map[key]
update the value of the node
move node to the front of the linked list (most recently used)
ELSE:
    IF cache is full:
        remove the least recently used node from the linked list (tail)
        remove the key of the least recently used node from the hash map
    CREATE a new node with the given key and value
    ADD the new node to the front of the linked list (most recently used)
    ADD the key-node pair to the hash map
FUNCTION move_to_front(node):
    REMOVE node from its current position in the linked list
    INSERT node at the front of the linked list
FUNCTION remove_node(node):
    IF node.prev exists:
        set node.prev.next to node.next
    IF node.next exists:
        set node.next.prev to node.prev
    IF node is the head of the linked list:
        set head to node.next
    IF node is the tail of the linked list:
        set tail to node.prev
    REMOVE node from memory (optional, depending on implementation)
FUNCTION add_to_front(node):
    SET node.next to the current head of the linked list
    SET the current head.prev to node (if head exists)
    SET head to node
    IF tail does not exist:
        set tail to node
```

Setelah peneliti, membuat pseudocode dari algoritma LRU peneliti juga membuat flowchart dari algoritma LRU. Berikut flowchart dari algoritma LRU.



Gambar 3.6 Flowchart Algoritma LRU

BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi Sistem

Untuk melakukan penelitian diperlukan perangkat keras, software, dan beberapa alat yang dibutuhkan untuk mendapatkan data - data agar hasil dari pengujian sistem dapat diimplementasikan.

4.1.1. *Software dan Hardware*

Adapun detail *software* dan *hardware* yang dipakai untuk melaksanakan penelitian ini meliputi.

1. Laptop-O35Q7B1M Asus *TUF Gaming F15*
2. *Smartphone* OPPO Reno 11 F 5G
3. *Processor Intel CoreTMi5-10300H Processor 2.5 GHZ (8M Cache, up to 4.5 GHZ, 4 cores)*
4. RAM 8GB
5. *Storage SSD 512GB*
6. Sistem Operasi *Windows 11 Home 64-bit*
7. Arduino IDE versi 2.3.2
8. Web ThingSpeak
9. Ubuntu versi 6
10. SD card 32 gb

4.1.2. Alat

Peralatan yang digunakan untuk penelitian ini sebagai berikut.

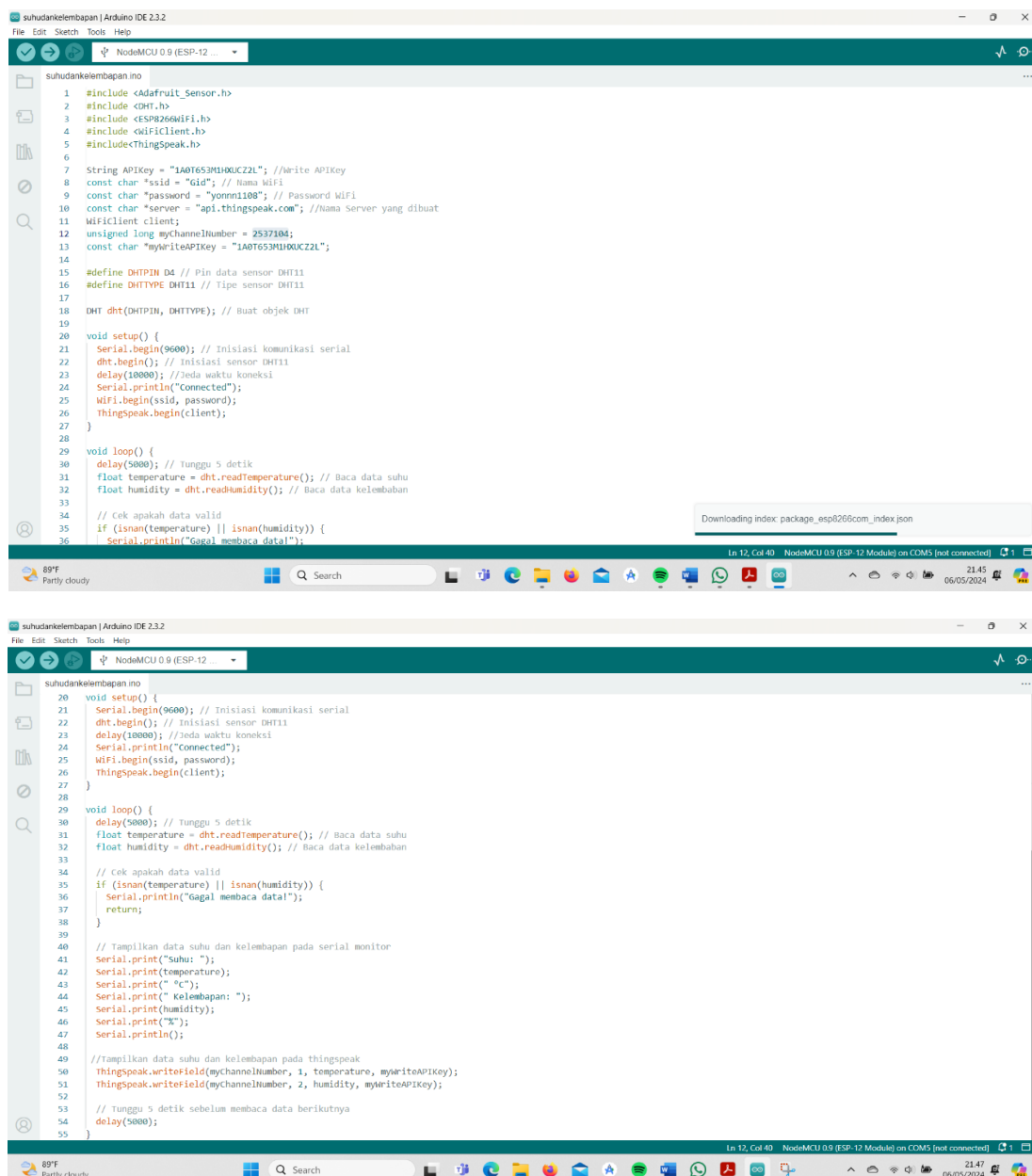
1. ESP8266
2. *Humidity sensor dht11*
3. Raspberry pi
4. *Cable Jumper female to female*
5. USB Cable

4.1.3. Tahap Pengerjaan

Tahap pengerjaan pada penelitian ini yang pertama dilakukan adalah menghubungkan pin pada humidity sensor dht11 ke pin esp8266, kemudian esp8266 dihubungkan ke laptop menggunakan usb cable agar esp8266 beserta sensor dht11 dapat terbaca di esp8266 sehingga program untuk sensor dht11 dapat dibuat. Berikut gambar dari tahap pengerjaan.

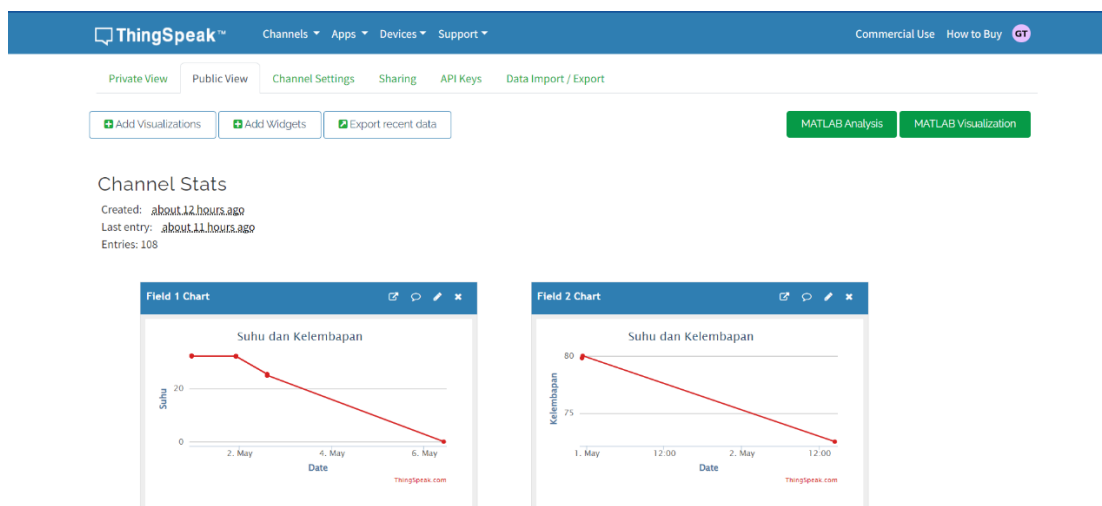


Gambar 4.1 Pin dht11 terhubung ke esp8266 kemudian dihubungkan ke laptop melalui usb cable



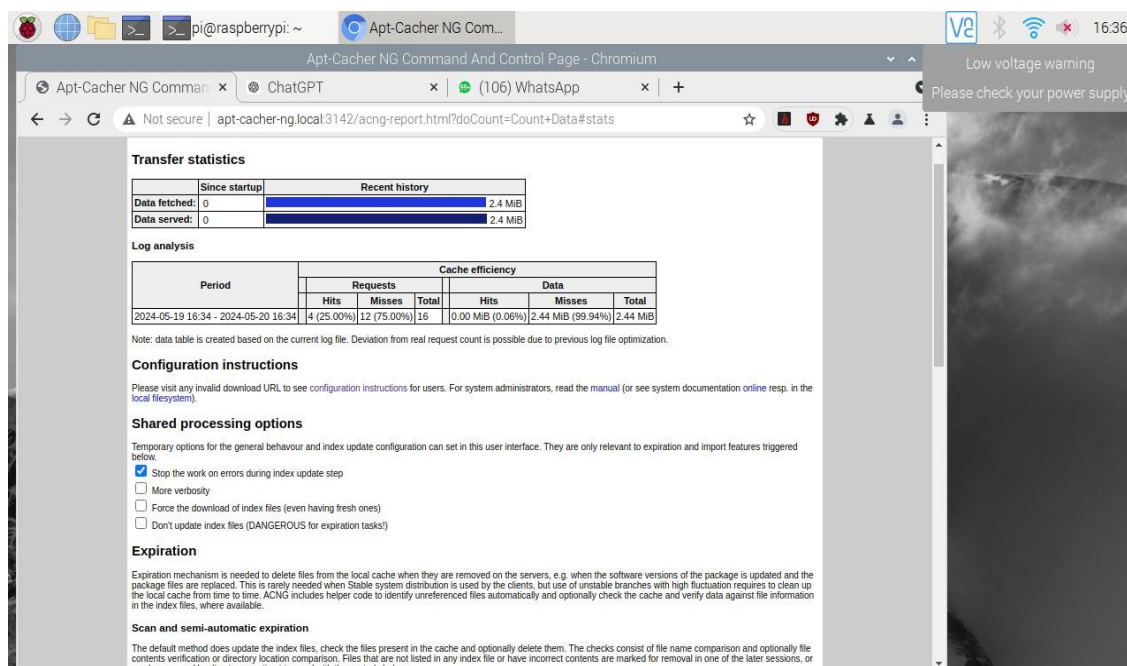
Gambar 4.2 Program untuk dht11

Setelah menyelesaikan rangkaian sensor dht11 dan esp8266 yang saling terhubung ke masing – masing pin, dan juga menyelesaikan program sensor dht11 untuk mendapatkan data suhu dan kelembapan selanjutnya peneliti memasukkan data ke web thingspeak agar data dari sensor dht11 dapat ditampilkan, setelah itu peneliti mengupload data tersebut ke website thingspeak agar client atau publik dapat melihat hasil dari data sensor dht11.



Gambar 4.3 Data suhu dan kelembapan pada thingspeak

Selanjutnya, peneliti melakukan konfigurasi untuk membuat tempat *cache* dari data sensor dht11 dengan *apt-cacher-ng* agar data yang dikirim ke client terlebih dahulu dikirim ke *apt-cacher-ng* sehingga client hanya mengakses data tanpa ada *cache* yang ikut terkirim. Berikut gambar dari *apt-cacher-ng* yang telah dibuat



Gambar 4.4 Penyimpanan cache dari data sensor

Setelah melakukan konfigurasi untuk membuat tempat *cache* dari data sensor, peneliti melakukan konfigurasi agar penyimpanan *cache* tersebut dapat melakukan *caching* untuk URL Thingspeak. Kemudian, peneliti juga melakukan konfigurasi *proxy* pada esp8266 agar memastikan data dari Thingspeak dapat dikirim melalui *proxy*. Salah satu

cara adalah dengan menggunakan orangepi sebagai gateway atau perantara yang bisa mengatur cache. Berikut adalah cara untuk mengatur raspberrypi sebagai perantara:

Langkah 1 Instalasi ‘apt-cacher-ng’ di Raspberrypi :

1. Update sistem dan install ‘apt-cacher-ng’ :

```
sudo apt-get update
```

```
sudo apt-get install apt-cacher-ng
```

2. Mulai layanan ‘apt-cacher-ng’ :

```
sudo systemctl start apt-cacher-ng
```

```
sudo systemctl enable apt-cacher-ng
```

3. Edit file konfigurasi ‘/etc/apt-cacher-ng/acng.conf’ untuk hanya menyimpan cache dari jenis file tertentu (HTML, CSS, dan JavaScript).

```
sudo nano /etc/apt-cacher-ng/acng.conf
```

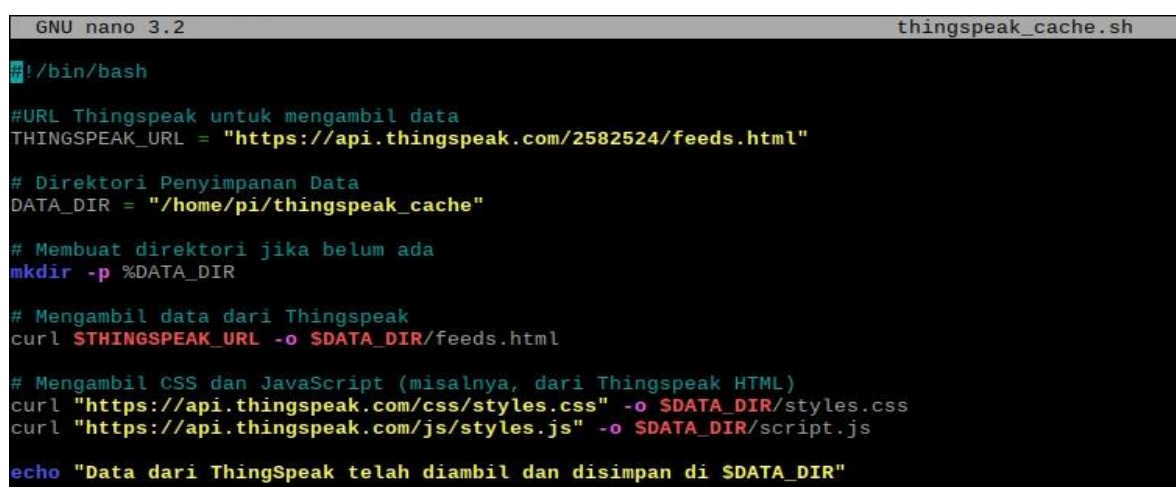
Tambahkan atau sesuaikan konfigurasi berikut :

```
PassThroughPattern: .*\. (html|css|js)$
```

Pada langkah pertama, peneliti melakukan instalasi ‘apt-cacher-ng’ yang bertujuan untuk menyimpan cache dan melihat efisiensi cache yang disimpan di dalam raspberrypi. Kemudian, peneliti melakukan konfigurasi di ‘/etc/apt-cacher-ng/acng.conf’ yang bertujuan agar cache yang diambil dan disimpan merupakan cache html, css, dan javascript. Setelah melakukan instalasi dan konfigurasi peneliti membuat skrip untuk mengambil data dari thingspeak sekaligus menyimpan cache.

Langkah 2 Buat Skrip Untuk Mengambil Data dari ThingSpeak :

Buat skrip ‘thingspeak_cache.sh’:



```
GNU nano 3.2 thingspeak_cache.sh
#!/bin/bash

#URL Thingspeak untuk mengambil data
THINGSPEAK_URL = "https://api.thingspeak.com/2582524/feeds.html"

# Direktori Penyimpanan Data
DATA_DIR = "/home/pi/thingspeak_cache"

# Membuat direktori jika belum ada
mkdir -p %DATA_DIR

# Mengambil data dari Thingspeak
curl $THINGSPEAK_URL -o $DATA_DIR/feeds.html

# Mengambil CSS dan JavaScript (misalnya, dari Thingspeak HTML)
curl "https://api.thingspeak.com/css/styles.css" -o $DATA_DIR/styles.css
curl "https://api.thingspeak.com/js/styles.js" -o $DATA_DIR/script.js

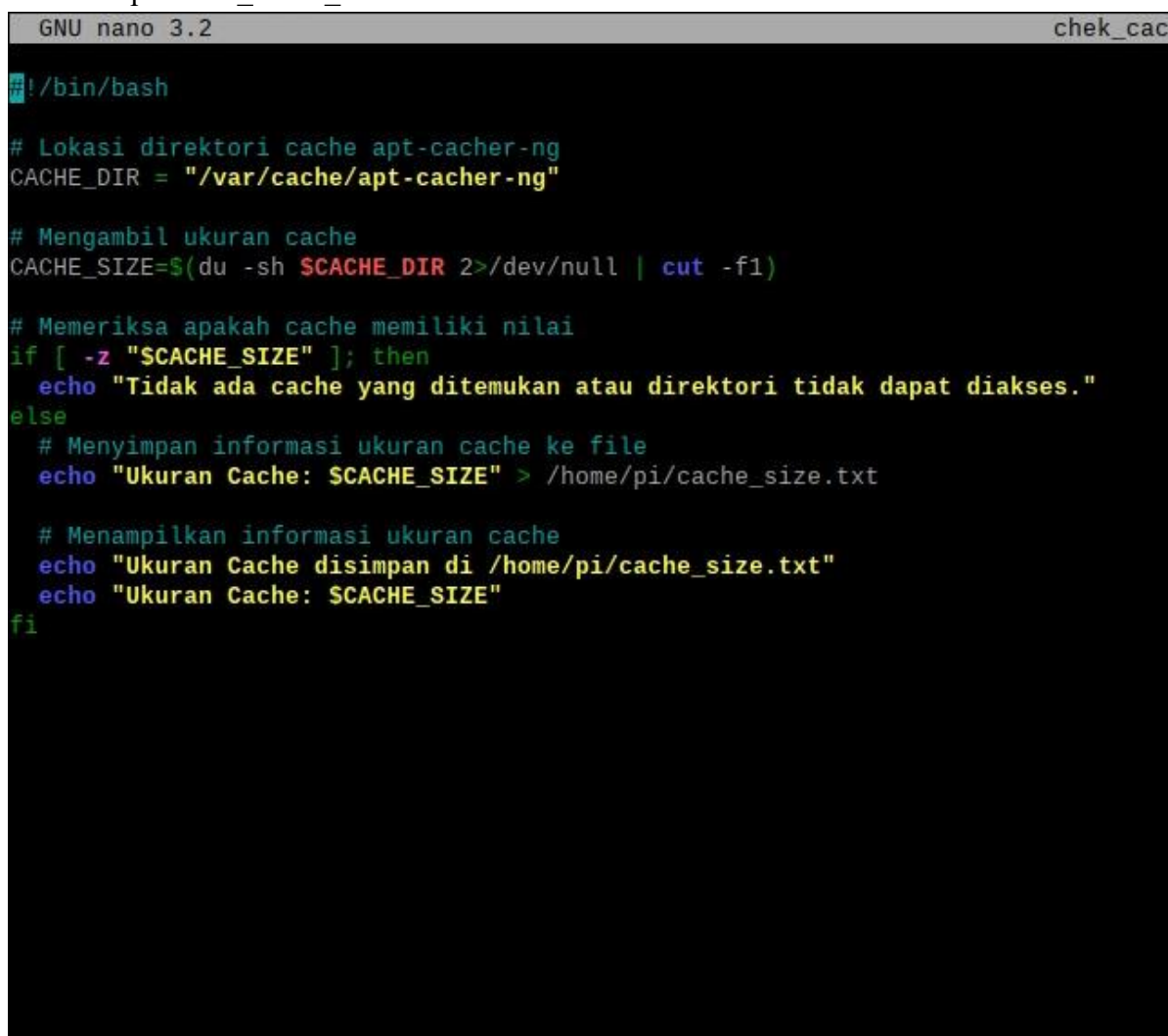
echo "Data dari ThingSpeak telah diambil dan disimpan di $DATA_DIR"
```

Gambar 4.5 Skrip Menyimpan Cache Thingspeak

Pada gambar 4.5 merupakan skrip untuk menyimpan cache yang berasal dari thingspeak yang bertujuan agar cache yang ada pada data di thingspeak dapat disimpan di dalam apt-cacher-ng yang berada di raspberrypi. Selanjutnya peneliti membuat skrip untuk melihat ukuran cache.

Langkah 3 Buat Skrip Untuk Melihat Ukuran Cache :

Buat skrip 'check_cache_size.sh' :



```
GNU nano 3.2 chek_cac

#!/bin/bash

# Lokasi direktori cache apt-cacher-ng
CACHE_DIR = "/var/cache/apt-cacher-ng"

# Mengambil ukuran cache
CACHE_SIZE=$(du -sh $CACHE_DIR 2>/dev/null | cut -f1)

# Memeriksa apakah cache memiliki nilai
if [ -z "$CACHE_SIZE" ]; then
    echo "Tidak ada cache yang ditemukan atau direktori tidak dapat diakses."
else
    # Menyimpan informasi ukuran cache ke file
    echo "Ukuran Cache: $CACHE_SIZE" > /home/pi/cache_size.txt

    # Menampilkan informasi ukuran cache
    echo "Ukuran Cache disimpan di /home/pi/cache_size.txt"
    echo "Ukuran Cache: $CACHE_SIZE"
fi
```

Gambar 4.6 Skrip Untuk Melihat Ukuran Cache

Setelah membuat skrip untuk melihat ukuran cache yang sudah disimpan di raspberrypi, peneliti selanjutnya membuat skrip algoritma LRU yang bertujuan untuk membersihkan cache yang sudah lama tidak digunakan.

Langkah 4 Buat Skrip Algoritma LRU Untuk Membersihkan Cache

```
File Edit Tabs Help
GNU nano 3.2 lru_cache_algorithm.sh

#!/bin/bash

# Lokasi direktori cache apt-cacher-ng
CACHE_DIR="/var/cache/apt-cacher-ng"

# Batas ukuran cache dalam gigabyte
CACHE_LIMIT=$((10 * 1024 * 1024 * 1024))

# Fungsi untuk mendapatkan ukuran direktori cache
get_cache_size() {
    du -sb $CACHE_SIZE | cut -f1
}

# Fungsi untuk membersihkan cache berdasarkan LRU
cache_algorithm() {
    echo "Membersihkan cache berdasarkan LRU..."

    # Loop sampai ukuran cache di bawah batas
    while [ $(get_cache_size) -gt $CACHE_LIMIT ]; do
        # Temukan file paling lama diakses
        OLDEST_FILE=$(find $CACHE_DIR -type f -printf '%A@ %p\n' | sort -n | head -n 1 | cut -d' ' -f2-)

        # Hapus file tersebut
        if [ -n "$OLDEST_FILE" ]; then
            echo "Menghapus file: $OLDEST_FILE"
            rm -f "$OLDEST_FILE"
        else
            echo "Tidak ada file yang ditemukan untuk dihapus"
            break
        fi
    done

    echo "Membersihkan Cache Selesai"
}

[ Read 41 lines ]
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify       ^C Cur Pos      M-U
^X Exit          ^R Read File     ^_ Replace       ^U Uncut Text    ^T To Spell      ^_ Go To Line    M-E
```

```
File Edit Tabs Help
GNU nano 3.2 lru_cache_algorithm.sh

#!/bin/bash

# Lokasi direktori cache apt-cacher-ng
CACHE_DIR="/var/cache/apt-cacher-ng"

# Batas ukuran cache dalam gigabyte
CACHE_LIMIT=$((10 * 1024 * 1024 * 1024))

# Fungsi untuk mendapatkan ukuran direktori cache
get_cache_size() {
    du -sb $CACHE_SIZE | cut -f1
}

# Fungsi untuk membersihkan cache berdasarkan LRU
cache_algorithm() {
    echo "Membersihkan cache berdasarkan LRU..."

    # Loop sampai ukuran cache di bawah batas
    while [ $(get_cache_size) -gt $CACHE_LIMIT ]; do
        # Temukan file paling lama diakses
        OLDEST_FILE=$(find $CACHE_DIR -type f -printf '%A@ %p\n' | sort -n | head -n 1 | cut -d' ' -f2-)

        # Hapus file tersebut
        if [ -n "$OLDEST_FILE" ]; then
            echo "Menghapus file: $OLDEST_FILE"
            rm -f "$OLDEST_FILE"
        else
            echo "Tidak ada file yang ditemukan untuk dihapus"
            break
        fi
    done

    echo "Membersihkan Cache Selesai"
}

[ Read 41 lines ]
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify       ^C Cur Pos      M-U
^X Exit          ^R Read File     ^_ Replace       ^U Uncut Text    ^T To Spell      ^_ Go To Line    M-E
```

Gambar 4.7 Skrip Untuk Algoritma LRU

Skrip algoritma LRU pada gambar 4.7 berfungsi untuk melakukan cache replacement agar penyimpanan cache yang ada di raspberry efektif.

Langkah 5 Memberikan Izin Eksekusi pada Semua Skrip

Setelah peneliti membuat semua skrip yang diperlukan, peneliti memberikan izin semua skrip agar bisa dijalankan. Berikut perintah untuk eksekusi semua skrip yang telah dibuat :

```
chmod +x /home/pi/thingspeak_data.sh
```

```
chmod +x /home/pi/chek_cache_size.sh
```

```
chmod +x /home/pi/lru_cache_algorithm.sh
```

Langkah 6 Menjalankan Skrip

Untuk menjalankan skrip dapat digunakan dengan 2 cara yaitu dengan menggunakan perintah 'crontab -e' dan mengatur kapan skrip akan dieksekusi ataupun dijalankan, dan yang kedua skrip di jalankan secara manual. Berikut perintah untuk menjalankan skrip secara manual :

- a. Jalankan skrip untuk mengambil data

```
/home/pi/thingspeak_data.sh
```

- b. Jalankan skrip untuk memeriksa ukuran cache :

```
/home/pi/chek_cache_size.sh
```

- c. Jalankan skrip untuk membersihkan cache :

```
sudo /home/pi/lru_cache_algorithm.sh
```

Langkah 7 Verifikasi Hasil

Untuk melihat hasil dari skrip tersebut, dapat menggunakan ketiga perintah berikut.

1. Periksa Ukuran Data Yang Diambil

```
du -sh /home/pi/thingspeak_data/*
```

2. Periksa Ukuran Cache Yang Disimpan di apt-cacher-ng

```
cat /home/pi/cache_size.txt
```

3. Verifikasi Ukuran Cache Secara Manual

```
du -sh /var/cache/apt-cacher-ng
```

4.2. Pengujian Ukuran Data

4.2.1 Ukuran Data Tanpa Ada *Proxy* Server

Berikut hasil ukuran data dari pengiriman sensor dht11 ke web server tanpa ada *proxy* server yang berfungsi untuk menyimpan cache. Peneliti melakukan 3 kali pengambilan data dan pengambilan data dilakukan selama 30 menit dengan interval yaitu 20 detik. Pengambilan data pertama dilakukan dengan interval setiap 20 detik selama 30 menit, di mana setiap permintaan data (*request size*) dan ukuran data yang diambil diproses secara terpisah sesuai dengan jeda waktu yang telah ditetapkan. Data dapat dilihat pada tabel 4.1

Tabel 4.1 Permintaan data dan ukuran data hari pertama

Lama Pengukuran (jam)	Permintaan Ukuran Data (byte)	Ukuran Data (kilobyte)
1	258	11.57
2	258	42.98
3	258	88.99
4	258	150.21

Pada tabel 4.1 dilakukan pengambilan data beserta ukuran data di hari pertama, dan dapat kita lihat bahwa semakin lama pengambilan data dilakukan maka semakin besar ukuran data yang diterima oleh *client*. Kemudian pengambilan data dilakukan kembali di hari kedua, dimana data yang sudah ada di hari pertama tetap disimpan. Untuk pengambilan data di hari kedua dapat dilihat pada tabel 4.3 berikut.

Tabel 4.2 Permintaan data dan ukuran data hari kedua

Lama Pengukuran (jam)	Permintaan Ukuran Data (byte)	Ukuran Data (kilobyte)
1	258	166.09
2	258	201.61
3	258	247.04
4	258	261.52

Setelah peneliti melakukan pengambilan data di hari kedua, dapat kita lihat pada tabel 4.3 bahwa ukuran data yang didapat semakin banyak, hal tersebut terjadi karena semakin lama pengambilan data dilakukan maka semakin besar ukuran data

yang diterima oleh *client*. Untuk pengambilan data di hari ketiga dapat dilihat pada tabel 4.4 berikut.

4.2.2 Ukuran Data Sebelum Menerapkan Algoritma LRU

Berikut hasil ukuran data yaitu berupa log analysis sebelum menerapkan Metode Cache Replacement dengan algoritma LRU, namun data dan cache telah dipisah. Peneliti menyajikan data dengan tabel.

Tabel 4.3 Perbandingan Efisiensi Cache dan Data Sebelum Menerapkan Metode Cache Replacement

Tanggal	Cache Efficiency					
	Requests			Data		
	Hits	Misses	Total	Hits	Misses	Total
2024 – 05 – 19 16:34 – 2024 – 05 – 20 16:34	4 (25.00%)	12 (75.00%)	16	0.00 mb (0.06%)	3 mb (99.94%)	3 mb
2024 – 05 – 21 15 : 15 – 2024 – 05 – 22 15:15	3 (20.00%)	12 (80.00%)	15	0.5 mb (17.18%)	2.4 mb (82.82%)	2.4 mb
2024 – 06 – 30 10:48 – 2024 – 07 – 01 10:48	2 (18.18%)	9 (81.82%)	11	0.00 mb (0.02%)	14.28 mb (99.98%)	14.29 mb

Pada tabel 4.3 dapat dilihat perbandingan cache sebelum menerapkan Metode Cache Replacement dengan Algoritma LRU. Pengambilan cache dilakukan 3 hari. Terdapat 2 Efisiensi Cache yang didapatkan yaitu Efisiensi Cache *requests* dan data. Efisiensi Cache *requests* pada hari pertama, hari kedua dan hari ketiga terdapat perbedaan. Pada hari pertama Efisiensi Cache untuk *requests* jumlah *cache hits* yaitu 4 dengan persentase 25% dan jumlah *cache misses* yaitu 12 dengan persentase 91.67% dan total *requests* sebanyak 16. Sementara itu, Efisiensi Cache untuk data jumlah *cache hits* berjumlah 0.00 mb dengan persentase 0.06% dan jumlah *cache miss* 3 mb dengan persentase 99.94% dan total data yaitu 3 mb. Pada hari kedua Efisiensi Cache untuk *requests* jumlah *cache hits* yaitu 3 dengan persentase 20%, dan *cache misses* yaitu 12 dengan persentase 80% dan total *requests* yaitu 15. Sementara itu, Efisiensi Cache untuk data jumlah *cache hits* berjumlah 0.5 mb dengan persentase 17.18% dan jumlah *cache misses* yaitu 2.4 mb dengan persentase 82.82% dan total data yaitu 2.4 mb. Pada hari ketiga Efisiensi Cache untuk *requests* jumlah *cache hits* yaitu 2 dengan persentase 18.18% dan

jumlah *cache misses* yaitu 9 dengan persentase 81.82% dan total *requests* sebanyak 11. Sementara itu, Efisiensi Cache untuk data jumlah *cache hits* berjumlah 0.00 mb dengan persentase 0.02% dan jumlah *cache miss* 14.28 mb dengan persentase 99.98% dan total data yaitu 14.29 mb. Dari data *cache* tersebut dapat kita lihat bahwa di hari pertama, hari kedua, dan hari ketiga lebih banyak *cache misses* dibandingkan dengan *cache hit*.

4.2.3 Ukuran Data Setelah Menerapkan Algoritma LRU

Berikut hasil ukuran data yaitu berupa log analysis setelah menerapkan Metode Cache Replacement dengan algoritma LRU, namun data dan cache telah dipisah. Peneliti menyajikan data dengan tabel.

Tabel 4.4 Perbandingan Efisiensi Cache dan Data Sesudah Menerapkan Metode Cache Replacement

Period	Cache Efficiency					
	Requests			Data		
	Hits	Misses	Total	Hits	Misses	Total
2024 – 05 – 21 13 : 55 – 2024 – 05 – 22 13 : 55	1 (8.33%)	11 (91.67%)	12	0.25 mb (9.46%)	2.4 mb (90.54%)	2.7 mb
2024 – 05 – 22 13 : 55 – 2024 – 05 – 23 13 : 55	2 (66.67%)	1 (33.33%)	3	0.25 mb (94.3%)	0.01 mb (5.67%)	0.26 mb
2024 – 06 – 30 10:48 – 2024 – 07 – 01 10:48	6 (70.00%)	2 (30.00%)	8	11.48 mb (94.5%)	0.67 mb (5.54%)	12.15 mb

Pada tabel 4.4 dapat dilihat perbandingan cache setelah menerapkan Metode Cache Replacement dengan Algoritma LRU. Pengambilan cache juga dilakukan 3 hari. Terdapat 2 Efisiensi Cache yang didapatkan yaitu Efisiensi Cache *requests* dan data. Efisiensi Cache *requests* pada hari pertama dan hari kedua terdapat perbedaan. Pada hari pertama Efisiensi Cache untuk *requests* jumlah *cache hits* yaitu 1 dengan persentase 8.33% dan jumlah *cache misses* yaitu 11 dengan persentase 91.67% dan total *requests* sebanyak 12. Sementara itu, Efisiensi Cache untuk data jumlah *cache hits* berjumlah 0.25 mb dengan persentase 9.46% dan jumlah *cache miss* 2.4 mb dengan persentase 90.54% dan total data yaitu 2.7 mb. Pada hari kedua Efisiensi Cache untuk *requests* jumlah *cache hits* yaitu 2 dengan persentase 66.67%, dan *cache misses* yaitu 1 dengan persentase 33.33% dan total *requests* yaitu 3. Sementara itu, Efisiensi Cache

untuk data jumlah *cache hits* berjumlah 0.25 mb dengan persentase 94.3% dan jumlah *cache misses* yaitu 0.01 mb dengan persentase 5.67% dan total data yaitu 0.26 mb. Pada hari ketiga Efisiensi Cache untuk *requests* jumlah *cache hits* yaitu 6 dengan persentase 70.00%, dan *cache misses* yaitu 2 dengan persentase 30.00% dan total *requests* yaitu 2. Sementara itu, Efisiensi Cache untuk data jumlah *cache hits* berjumlah 11.48 mb dengan persentase 94.5% dan jumlah *cache misses* yaitu 0.67 mb dengan persentase 5.54% dan total data yaitu 12.15 mb. Walaupun pada hari pertama terdapat *cache miss* yang lebih banyak untuk *requests* dan data namun, di hari kedua, dan di hari ketiga *cache miss* yang didapat lebih sedikit hal tersebut juga terjadi untuk *cache hits*.

Berdasarkan analisis pada Tabel 4.3 dan Tabel 4.4, dapat disimpulkan bahwa sebelum menerapkan Metode Cache Replacement dengan Algoritma LRU, jumlah *cache miss* lebih banyak dibandingkan dengan *cache hit* selama tiga hari pengukuran. Pada hari pertama, kedua, dan ketiga, Efisiensi Cache untuk *requests* dan data menunjukkan persentase *cache miss* yang tinggi, dengan persentase *cache hit* yang rendah. Namun, setelah menerapkan Metode Cache Replacement dengan Algoritma LRU, terjadi peningkatan signifikan dalam efisiensi *cache*. Meskipun pada hari pertama setelah penerapan masih terdapat *cache miss* yang lebih banyak untuk *requests* dan data, pada hari kedua dan ketiga, jumlah *cache miss* berkurang secara signifikan dan jumlah *cache hit* meningkat. Hal ini menunjukkan bahwa Algoritma LRU berhasil meningkatkan kinerja *cache*, mengurangi jumlah *cache miss*, dan meningkatkan efisiensi penggunaan data yang di-*cache*.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan proses penelitian yang sudah dilakukan diperoleh hasil sebagai berikut:

1. Data suhu dan kelembapan bisa diakses oleh semua orang.
2. Algoritma LRU pada metode cache replacement berhasil diterapkan untuk efektifitas penyimpanan pada jaringan IoT.
3. Metode Cache Replacement algoritma LRU mampu mengurangi hit miss pada cache.
4. Metode Cache Replacement LRU efektif digunakan untuk efektivitas penyimpanan cache pada jaringan iot hingga 90%.

5.2 Saran

Penelitian ini tentu saja memiliki kekurangan, untuk itu peneliti memberikan saran yang mampu diberikan agar penelitian ini lebih baik ke depannya, beberapa saran yang diberikan antara lain:

1. Menambahkan skrip pre-fetching cache yang bertujuan untuk mengurangi cache miss ketika ingin mendapatkan cache dari data sensor dan juga dapat mengoptimalkan pola permintaan data dari client sehingga jika client meminta data dalam interval, skrip pre-fetching dapat diatur sesuai dengan interval tersebut.
2. Menggunakan metode cache replacement lainnya seperti metode cache replacement LFU, FIFO, dan beberapa metode cache replacement yang lain.
3. Melakukan analisis workload berdasarkan profiling data access patterns yang bertujuan untuk menganalisis pola akses data yang lebih rinci untuk melihat apakah ada pola khusus yang dapat dioptimalkan lebih lanjut kemudian dapat melakukan simulation of different workloads yang bertujuan untuk menguji algoritma dengan berbagai skenario beban kerja (workload) untuk memastikan bahwa algoritma bekerja dengan baik dalam berbagai situasi.

DAFTAR PUSTAKA

- Aisyah, A. (2020). *RANCANG BANGUN REAL-TIME MESSAGING PROTOCOL UNTUK APLIKASI LIVE STREAMING BERBASIS CLOUD SERVER*
<https://doi.org/10.13140/RG.2.2.19106.66241>
- APJII (Asosiasi Penyedia Jasa Internet Indonesia). 2023. Survei Internet Indonesia 2023 Tahap 1. *Press Conference*. (Online) www.survei.apjii.or.id.
 (9 Januari 2024)
- Ayu, M. G. (2020, Oktober 17). Perkembangan dan penggunaan IoT di Indonesia. Cloud Computing Indonesia.
<https://www.cloudcomputing.id/berita/perkembangan-dan-penggunaan-iot-di-indonesia>
- Afriansyah, D., & Kurniawan, D. F. (2022). Membangun File Transfer Protocol (FTP) Server dengan Debian pada Smks Budi Utama Pajaresuk Kabupaten Pringsewu. *Jurnal Informatika Software dan Network (JISN)*, 3(2).
- Al - Hamdani, A. A., Al - Omari, A. A., & Al - Rifai, A. A. (2022). The Role of WAN in IoT Applications. *Journal of Network and Computer Applications*, 102-117.
- Al Azizi, M. N. R., Negara, R. M., & Yovita, L. V. (2023). Analisis Pengaruh Jumlah Konten Dan Node Dengan Cache Replacement Lru Pada Virtual Node NDN. *eProceedings of Engineering*, 9(6).
- Chandra, A. Y. (2019). Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request. *Jurnal Sistem Dan Informatika (JSI)*, 14(1), 48-56. <https://doi.org/10.30864/jsi.v14i1.248>
- Corral-Plaza, D., Medina-Bulo, I., Ortiz, G., & Boubeta-Puig, J. (2020). A stream processing architecture for heterogeneous data sources in the Internet of Things. *Computer Standards & Interfaces*, 70, 103426.
- Costa, N., Jayasinghe, M., Atukorale, A., Abeysinghe, S., Perera, S., & Perera, I. (2019, June). Adapt-t: An adaptive algorithm for auto-tuning worker thread pool size in application servers. In *2019 IEEE Symposium on Computers and Communications (ISCC)* (pp. 1-6). IEEE.

- Da Silva, E. T., De Macedo, J. M. H., & Costa, A. L. D. (2022). NDN Content Store and Caching Policies: Performance Evaluation. *Computers, 11*(3), 37. <https://doi.org/10.33990/computers11030037>.
- Dwiyatno, S., Rachmat, E., Sari, A. P., & Gustiawan, O. (2020). Implementasi Virtualisasi Server Berbasis Docker Container. *PROSISKO: Jurnal Pengembangan Riset Dan Observasi Sistem Komputer, 7* (2), 165–175.
- Elsaleh, T., Bermudez-Edo, M., Enshaeifar, S., Acton, S. T., Rezvani, R., & Barnaghi, P. (2019, June). IoT-stream: a lightweight ontology for internet of things data streams. In *2019 Global IoT Summit (GloTS)* (pp. 1-6). IEEE.
- Fathulrohman, Y. N. I., & Saepulloh, A. (2019). Alat Monitoring suhu dan kelembaban menggunakan arduino uno. *Jurnal Manajemen dan Teknik Informatika (JUMANTAKA), 2*(1).
- Gunawan, A. A. N., & Sumadiyasa, M. (2019). Water Level Detection System Based on Ultrasonic Sensors HC-SR04 and ESP8266-12 Modules with Telegram and Buzzer Communication Media. *Instrumentation, Measures, Métrologies, 18*(3).
- Hawari, M. A., & Prastowo, B. N. (2021). Analisa Karakteristik Single Board Computer sebagai Streaming Video Server. *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems), 11*(2), 179-188.
- Hadi, S., Labib, R. P. M. D., & Widayaka, P. D. (2022). Perbandingan Akurasi Pengukuran Sensor LM35 dan Sensor DHT11 untuk Monitoring Suhu Berbasis Internet of Things. *STRING (Satuan Tulisan Riset dan Inovasi Teknologi), 6*(3), 269-278.
- Jaelani, A. K., & Rosyid, H. IMPLEMNTASI APPLICATION-LEVEL CACHE UNTUK OPTIMASI KINERJA WEB SERVER (“STUDI KASUS SIM SURAT PT. GRESIK MIGAS”).
- Kalsum, T. U., & Riska, R. (2022). Penerapan PI Hole DNS Server Sebagai ADS Blocker Dan Sistem Filtering Website Pada Jaringan Hotspot. *JURNAL MEDIA INFOTAMA, 18*(2), 208-217.
- Kenda, K., Kažič, B., Novak, E., & Mladenčić, D. (2019). Streaming data fusion for the internet of things. *Sensors, 19*(8), 1955.
- Muafani, M. (2020, December 19). PEMANFAATAN INTERNET OF THINGS (IOT) PADA DESAIN RUMAH TINGGAL. *Jurnal Ilmiah Arsitektur, 10*(2), 61-66. <https://doi.org/https://doi.org/10.32699/jiars.v10i2.1620>

- Michael, A., Hermawan, H., & Pratiwi, H. I. (2019). Sistem Monitoring Server Dengan Menggunakan SNMP. *Widyakala Journal*, 6(2), 163-166.
- Majid, M., Habib, S., Javed, A. R., Rizwan, M., Srivastava, G., Gadekallu, T. R., & Lin, J. C. W. (2022). Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review. *Sensors*, 22(6), 2087.
- Mukhtar, H., Sitorus, D. A. P., & Fatma, Y. (2020). Analisa Dan Implementasi Security Mail Server. *Jurnal Fasikom*, 10(1), 25-32.
- Nasir, A., & Lordianto, R. L. (2023). IMPLEMENTASI PROXY SERVER UNTUK OPTIMALISASI MANAJEMEN BANDWIDTH JARINGAN KOMPUTER PADA UNIVERSITAS XYZ. *Jurnal Technopreneur (JTech)*, 11(1), 16-23.
- Noviansyah, M., & Saiyar, H. (2020). Pemanfaatan Web Proxy Sebagai Pengoptimal Keamanan Jaringan Wireless Lan. *Jurnal Khatulistiwa Informatika*, 8(1).
- Pambudiyatno, N., Rifai, M., & Harianto, B. B. (2021). DESAIN JARINGAN INTERNET OF THINGS (IOT) DI POLITEKNIK PENERBANGAN SURABAYA. *Jurnal Penelitian*, 6(2), 90 – 100. <https://doi.org/10.46491/jp.v6i2.559>
- Nurjannah, I., Basuki, A., & Amron, K. (2019). Evaluasi Kinerja Strategi In-Network Caching pada Information-Centric Networking menggunakan Simulator Icarus. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(9), 9076-9084.
- Pratama, G., Mulyana, J., & Kusdiawan, W. (2020). ANALISIS DAN IMPLEMENTASI ALGORITMA LRU DAN GDSF SEBAGAI ATURAN CHASE REPLACEMENT PADA PROXY SERVER. *Syntax : Jurnal Informatika*, 9(2), 98 – 109. <https://doi.org/10.35076/syji.v9i2.3823>
- Pederson, M., Fitria, N., Sari, R. E., Yanti, Z. 2023. Implementasi DNS Server pada Sistem Operasi Ubuntu Menggunakan VirtualBox. *Journal of Network and Computer Applications* 2 : 52 – 62.
- Panduman, Y. Y. F., Funabiki, N., Ito, S., Husna, R., Kuribayashi, M., Okayasu, M., Shimazu, J., & Sukaridhoto, S. (2023). An Edge Device Framework in SEMAR IoT Application Server Platform. *Information*, 14(6), 312.

- Ramadhan, R. F., & Mukhaiyar, R. (2020). Penggunaan Database Mysql dengan Interface PhpMyAdmin sebagai Pengontrolan Smarthome Berbasis Raspberry Pi. *JTEIN: Jurnal Teknik Elektro Indonesia*, 1(2), 129-134.
- Rizal, C., Supiyandi, S., Zen, M., & Eka, M. (2022). Perancangan Server Kantor Desa Tomuan Holbung Berbasis Client Server. *Bulletin of Information Technology (BIT)*, 3(1), 27-33.
- Sadrif, I. N., & Afrianto, I. Penerapan Cloud Computing Pada Game Untuk Mengoptimalkan Performa.
- Salsabila, N., Choir, R. A., Tiara, S. I. N. J., Rahmadinanti, M. O., Fadah, H. I., Maryani, M., & Harijanto, A. (2023). RANCANG ALAT PRAKTIKUM UNTUK MENGUKUR SUHU MENGGUNAKAN SENSOR DS18B20 BERBASIS ARDUINO UNO. *Jurnal Sains Riset*, 13(2), 409-418.
- Santos, L., Costa, T., Caldeira, J. M., & Soares, V. N. (2022). Performance assessment of esp8266 wireless mesh networks. *Information*, 13(5), 210.
- Seo, J. (2020). Adaptable Online Game Server Design. *Journal of information and communication convergence engineering*, 18(2), 82-87.
- Suadi, W., Djanali, S., Wibisono, W., Anggoro, R., & Shiddiqi, A. M. (2020). GCRFP - PAGE REPLACEMENT FOR SOLID STATE DRIVE USING GHOST-CACHE. *Juti/JUTI (Jurnal Ilmiah Teknologi Informasi)*, 18(2), 94.
<https://doi.org/10.12962/j24068535.v18i2.a986>
- Sutikno, T., Satrian Purnama, H., Pamungkas, A., Fadlil, A., Mohd Alsofyani, I., & Jopri, M. H. (2021, December 1). Internet of things-based photovoltaics parameter monitoring system using NodeMCU ESP8266. *International Journal of Electrical and Computer Engineering (IJECE)*, 11(6), 5578.
<https://doi.org/10.11591/ijece.v11i6.pp5578-5587>
- Tanwir, T., Parma, H., & Sri, W. (2021). Peningkatan kinerja jaringan dengan menggunakan multi-rule algorithm. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIK)*, 8(1), 69-76.
- Villamil, S., Hernández, C., & Tarazona, G. (2020). An overview of internet of things. *Telkomnika (Telecommunication Computing Electronics and Control)*, 18(5), 2320-2327.

- Wijaya, M. C. (2020). Algoritme penggantian cache proxy terdistribusi untuk meningkatkan kinerja server web. *Jurnal Teknologi dan Sistem Komputer*, 8(1), 1-5.
- Windiarti, Sri. (2020). SISTEM MANAJEMEN BASIS DATA. Artikel. *Jurnal Manajemen Sistem Informasi*.
- Wardhani, W., Hadi, S., & Budiarto, J. (2021). Rancang Bangun Sistem Monitoring Suhu dan Kelembaban Udara Pada Ruang Server Berbasis Wireless Sensor Network. *JTT (Jurnal Teknologi Terpadu)*, 9(2), 115-125.
- Wang, Y., Yang, Y., Han, C., Ye, L., Ke, Y., & Wang, Q. (2019). LR_LRU: A PACS - Oriented Intelligent Cache Replacement Policy. *IEEE Access*, 7, 58073 – 58084. <https://doi.org/10.1109/access.2019.2913961>
- Yovita, L. V., Wibowo, T. A., Ramadha, A. A., Satriawan, G. P., & Raniprma, S. (2022). Performance Analysis of Cache Replacement Algorithm using Virtual Named Data Network Nodes. *Jurnal Online Informatika*, 7(2), 203 – 210.