Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 9, September 2019, hlm. 9076-9084 http://j-ptiik.ub.ac.id

Evaluasi Kinerja Strategi *In-Network Caching* pada *Information-Centric Networking* menggunakan Simulator Icarus

Ismawati Nurjannah[1], Achmad Basuki[2], Kasyful Amron[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]ismajannah@student.ub.ac.id, [2]abazh@ub.ac.id, [3]kasyful@ub.ac.id

Abstrak

Information-centric networking adalah konsep arsitektur jaringan yang bersifat *content-centric model*, sehingga tidak perlu untuk terhubung langsung ke server agar mendapatkan konten yang dicari oleh *client*. Strategi *in-network caching* pada ICN merupakan strategi *caching* yang melakukan penyimpanan konten sebagai *cache* tersebar di dalam jaringan. *In-network caching* terdiri dua kombinasi strategi *cache placement* dan *cache replacement*. Strategi *cache placement* mengatur peletakan konten-konten pada kumpulan *router* yang bertugas sebagai *content store*, sedangkan strategi *cache replacement* mengatur penggantian konten ketika kapasitas penyimpanan *cache* penuh. Jika konten tidak dihapuskan secara cepat dan tepat maka akan berdampak pada efisiensi konten yang akan diterima oleh *client* sehingga mengakibatkan nilai *cache hit ratio* yang kecil dan meningkatnya latensi. Strategi *cache placement* dan *cache replacement* saling berkaitan satu sama lain dalam *in-network caching*. Konten perlu disebarkan ke jaringan melalui *content store* secara optimal serta menjamin bahwa konten yang disimpan merupakan yang terbaru dan berpeluang terjadi *cache hit*. Evaluasi kinerja strategi *in-network caching* dilakukan berdasarkan metrik kinerja *cache hit ratio*, latensi dan *path stretch*. Pengujian strategi *in-network caching* dilakukan dengan simulator Icarus yang dirancang khusus sesuai arsitektur ICN. Hasil yang didapatkan dari penelitian ini adalah kombinasi strategi *cache placement* ProbCache dengan *cache replacement* LFU memiliki kinerja yang lebih unggul sesuai parameter uji dibandingkan dengan strategi *in-network caching* lainnya. ProbCache dan LFU memperoleh rerata nilai CHR sebesar 54,02%, rerata latensi 0,54 ms, dan rerata rasio *path stretch* sebesar 4,76.

Kata kunci: *Information-centric networking, in-network caching, cache, Icarus*

Abstract

Information-centric networking is a concept of network architecture which is a content-centric model, so there is no need to connect directly to the server to get the content sought by the client. The in-network caching strategy in ICN is a caching strategy that stores content as cache spread across the network. In-network caching consists of two combinations of cache placement and cache replacement strategies. The cache placement strategy manages the placement of content on a collection of routers that acts as a content store, while the cache replacement strategy manages content replacement when the cache storage capacity is full. If the content is not removed quickly and precisely it will have an impact on the efficiency of the content that will be received by the client, resulting in a small cache hit ratio and increased latency. The cache placement and cache replacement strategies are interrelated with each other in in-network caching. Content needs to be disseminated to the network through the content store optimally and ensure that the stored content is the latest and has the chance of a cache hit. Performance evaluation of in-network caching strategies is based on cache hit ratio performance metrics, latency and path stretch. In-network caching strategy testing is done with the Icarus simulator which is specifically designed according to the ICN architecture. The results obtained from this study are the combination of ProbCache cache placement strategy with LFU cache replacement has a superior performance according to test parameters compared to other in-network caching strategies. ProbCache and LFU obtained an average CHR value of 54.02%, an average latency of 0.54 ms, and an average path stretch ratio of 4.76.

Keywords: *Information-centric networking, in-network caching, cache, Icarus*

1. PENDAHULUAN

Mayoritas lalu lintas Internet saat ini didominasi oleh permintaan dan penyebaran konten dari situs web populer seperti, Youtube dan Facebook. *Bandwidth* Internet menjadi terbuang begitu saja oleh pengunduhan konten yang identik secara berulang (Kim, et al., 2013). Selain itu, meskipun konten diunduh secara berulang, pengguna tetap akan meminta konten ke penyedia konten asli. *Information-centric Networking* (ICN) merupakan konsep arsitektur yang diusulkan untuk menjadi solusi permasalahan di atas. Secara umum, ICN merupakan *content-centric model*, yang artinya tidak diperlukan lagi untuk terhubung langsung ke *origin server* guna mendapatkan konten. Sebagai gantinya, pengguna dapat secara langsung mengirimkan permintaan mengenai konten yang diminta ke jaringan tanpa mempertimbangkan lokasi original konten tersebut.

ICN memiliki tiga struktur data, yakni *Pending Interest Table* (PIT), *Content Store* (CS) dan *Forwarding Information Base* (FIB). PIT berisikan paket-paket permintaan konten yang sedang dicari dan akan dikirim menuju kumpulan *router*. *Content Sto*re merupakan *router-router* yang menyimpan *cache* konten. Ketika konten yang dicari pada *router* berhasil ditemukan maka konten dapat langsung dikirimkan ke penerima tanpa harus terhubung dengan *origin server*. Namun, ketika konten yang dicari tidak ditemukan pada router-router di dalam jaringan tersebut, maka *Interest packet* yang berisi paket permintaan akan disimpan sementara pada PIT, dan selanjutnya diteruskan menuju *router* lainnya sesuai dengan informasi paket di FIB. Kapan pun ketika *Interest packet* berhasil menemukan konten yang dicari, maka informasi entri dari *Interest packet* akan dihapuskan dari PIT (Rossi, et al., 2011).

In-network caching merupakan metode penyimpanan *cache* pada arsitektur ICN. Strategi ini bertujuan untuk meningkatkan efisiensi jaringan dan kinerja distribusi konten dengan cara menyimpan *cache* pada *router-router* di jaringan (Yuemei Xu, 2016). Strategi pada *In-network caching* diharapkan mampu memperkecil PIT yang artinya penerima tidak mengalami kendala dalam menemukan konten yang diminta dalam waktu sesingkatnya dan memperbesar peluang *cache hit*. ICN memiliki dua strategi yang saling berkaitan di dalam *In-network caching*, yaitu strategi peletakan *cache* (*cache placement*) dan penggantian *cache* (*cache replacement*) (Jacobson, 2009).

Cache placement akan mengatur bagaimana sebuah konten diletakkan pada kumpulan *router* yang tersebar di jaringan. Peletakan konten pada tiap router perlu mempertimbangkan ukuran penyimpanan yang tersedia serta mempertimbangkan konten apa saja yang akan disimpan.

Cache replacement perlu dilakukan guna menjaga kebaruan konten pada *content store* serta memastikan bahwa konten bersifat beragam sehingga kapan pun ada *Interest packet* yang datang akan segera mendapatkan konten yang diinginkan serta mampu meminimalisir entri di PIT. Ketika konten tidak dihapuskan secara cepat dan tepat, maka akan berdampak pada efisiensi konten yang akan diterima pada pengguna sehingga mengakibatkan nilai *cache hit ratio* yang kecil dan meningkatnya latensi di sisi penerima. Strategi *cache placement* dan *cache replacement* saling berkaitan satu sama lain. Konten perlu disebarkan ke jaringan melalui *content store* secara optimal serta menjamin bahwa konten yang disimpan merupakan yang terbaru dan berpeluang terjadi *cache h*it.

Menurut penelitian yang dilakukan oleh Bernardini pada tahun 2016 menyebutkan bahwa strategi *cache placement* ProbCache dan LCE dengan *cache replacement* LRU memiliki performa yang baik dalam kondisi *low popularity* yang artinya tingkat keragaman konten yang rendah di dalam jaringan. Penelitian (Maulana, 2018) tentang kinerja *cache placement* antara *on-path caching* dan *off-path caching* menyebutkan bahwa performa yang baik berasal dari *on-path caching*, yakni ProbCache dengan strategi *cache replacement* LRU. Sedangkan, penelitian (Wang, et al., 2013) menunjukkan bahwa strategi *cache replacement* LFU memiliki perfoma yang lebih baik dibandingkan *cache replacement* yang lainnya. LFU disebut memiliki *hop savings* yang besar artinya mampu memotong jalur menjadi lebih pendek dibandingkan dengan jalur sesungguhnya dari *origin server* menuju penerima.

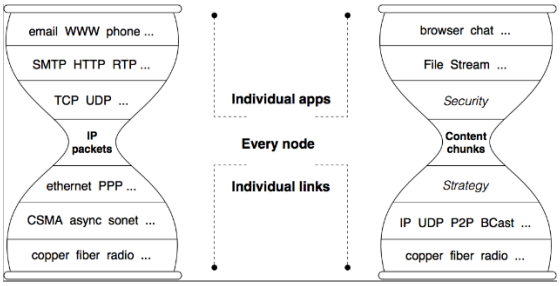
Pada penelitian-penelitian yang telah disebutkan di atas, menyatakan bahwa strategi *In-network caching* yang memiliki kinerja terbaik hanya fokus pada salah satu strategi, yaitu antara strategi *cache placement* dan strategi *cache replacement* tanpa

mempertimbangkan kombinasi di antara kedua strategi tersebut. Maka penelitian ini bertujuan untuk mengevaluasi kinerja *In-network caching* dengan mempertimbangkan kombinasi dari strategi *cache placement* dan *cache replacement* menggunakan simulator Icarus. Sehingga hasil dari penelitian ini dapat memberikan sudut pandang baru dalam memperbaiki kinerja *In-network caching* dengan membandingkan kombinasi strategi *cache placement* dan *cache replacement*. Selanjutnya, penelitian ini membandingkan kinerja dari kombinasi *cache placement*, yakni ProbCache dan LCE dengan *cache replacement* LRU dan LFU pada kondisi *low popularity* dan *high popularity* serta mengukur kinerja berdasarkan *Cache Hit Ratio* (CHR), Latensi, dan *Path Strech*.

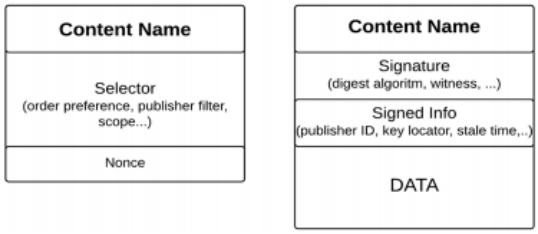
2. *INFORMATION-CENTRIC NETWORKING*

Information-Centric Networking (ICN) adalah sebuah paradigma jaringan baru untuk masa depan internet (Xylomenos, et al., 2014). Teknologi ICN menerapkan pendekatan *content-centric model* daripada teknologi saat ini yang masih menggunakan *host-centric model*. Kelebihan yang dimiliki oleh ICN adalah kemampuan pencarian data yang lebih cepat karena tiap *node* atau *router* mampu menyimpan *cache* sehingga permintaan akan paket data tidak perlu menuju ke *origin server* untuk mendapatkan konten yang diinginkan. Perbedaan yang mendasar mengenai ICN dengan teknologi TCP/IP saat ini adalah model komunikasi yang digunakan. TCP/IP berfokus pada komunikasi antar *host* sehingga pencarian konten akan terpaku pada letak *host* yang menyediakan sumber konten yang dicari, sedangkan pada ICN dilakukan *caching* dalam tiap *node*. *Node-node* yang berada pada jaringan tersebut menyimpan *cache* konten sehingga dalam pencarian konten tidak harus menuju *origin server* melainkan dapat ditemukan pada *node* yang terhubung di dalam jaringan.

Pada arsitektur ICN, data dibentuk menjadi sebuah rangkaian potongan-potongan (*data chunk*) yang disusun berdasarkan nama secara hirarki yang disebarkan oleh *routing protocol*. Adapun fitur yang disediakan oleh ICN adalah *content distribution*, *multicast*, *mobility* dan *delay-tolerant networking* (Wang, et al., 2012).



Gambar 1. Perbandingan *Layer* pada TCP/IP dan ICN
Sumber: Van Jacobson, et al. (2009)



Gambar 2. Paket *Interest* dan Paket Data pada ICN
Sumber: Van Jacobson, et al. (2009)

Terdapat dua jenis paket yang ditangani oleh ICN, yaitu *Interest packet* dan *Data packet*. *Interest packet* berisi permintaan atas konten yang dicari dan disebarkan pada jaringan, sedangkan *Data packet* yang merupakan hasil konten yang dicari tersebut (Jacobson, 2009). Jika dibandingkan dengan arsitektur TCP/IP, *Interest packet* memiliki fungsi yang sama seperti *paket request*.

ICN memiliki 3 struktur data, yakni *Pending Interest Table* (PIT), *Content Store* (CS) dan *Forwarding Information Base* (FIB). Pada PIT berisikan kumpulan paket permintaan konten yang sedang dicari dan dikirim menuju *router*. PIT juga berfungsi untuk mencegah duplikasi pengiriman *Interest packet* dan *Data packet* karena pada *node* akan mempertahankan setiap entri yang menyimpan nama konten pada *Interest Packet* dan kumpulan *interfaces* yang merupakan sumber *Interest Packet* yang diterima oleh pengguna (Kim & Yeom, 2013).

Content Store (CS) merupakan *node* yang menyimpan *cache* konten. Ketika konten yang dicari pada *node* ditemukan maka konten dapat langsung dikirimkan ke penerima tanpa harus terhubung dengan *origin server*. Namun, ketika konten yang dicari tidak ditemukan pada *node-node* di jaringan, maka *Interest packet* akan disimpan pada PIT dan diteruskan sesuai informasi paket di FIB. Kapan pun ketika *Interest packet* berhasil ditemukan, maka informasi entri yang sesuai dengan konten yang

dicari akan dihapuskan dari PIT (Rossi, et al., 2011).

3. *IN-NETWORK CACHING*

Caching adalah mekanisme untuk menyediakan penyimpanan sementara yang berguna menurunkan *bandwidth*, beban server dan waktu respon (Kim & Yeom, 2013). Di dalam arsitektur ICN, mekanisme penyimpanan konten berperan penting. *Node content source* memiliki ruang penyimpanan konten sehingga permintaan konten dapat dilayani oleh *content source* yang menyimpan salinan dari konten tersebut. *Caching* yang dilakukan oleh *content source* dapat menghindari pemborosan *network bandwidth* yang disebabkan oleh pengiriman konten populer secara berulang. Dari segi penerima dapat menurunkan waktu respon untuk mendapatkan konten yang dicari melalui penempatan konten yang lebih dekat dengan penerima (Kim & Yeom, 2013).

In-network caching adalah strategi *caching* yang dilakukan oleh kumpulan *router* yang terhubung pada sebuah jaringan. *Router* pada *in-network caching* berfungsi sebagai *routing content* sekaligus *saving content*. *Routing content* artinya *router* dapat meneruskan konten ke pengguna mau pun meneruskan ke *router* lainnya. *Saving content* artinya *router* dapat menyimpan konten selama ruang penyimpanan pada *content store* cukup. Secara umum, *in-network caching t*erdapat 2 strategi yang saling berkaitan, yaitu *cache placement* dan *cache replacement.*

Cache placement adalah strategi dalam meletakkan sebuah konten pada kumpulan *router* yang tersebar di jaringan. Sedangkan pada *cache replacement* mengatur penggantian konten pada ruang penyimpanan *router* agar konten yang disimpan merupakan konten yang terbaru dan memiliki peluang *cache hit* lebih besar.

Metrik pengujian yang digunakan untuk mengukur kinerja strategi *in-network caching* adalah:

- *Cache Hit Ratio* (CHR): Metrik ini merupakan perbandingan jumlah paket *interest* dan paket data. Paket data merupakan permintaan yang berhasil dilayani (*cache hit*). Adapun persamaan untuk menghitung CHR adalah:

$$CHR = \frac{n(hit)}{n(total)} \quad (1)$$

Pada n(hit) merepresentasikan jumlah permintaan yang berhasil dilayani oleh *content store* (cache hit) sedangkan pada n(total) merupakan jumlah total permintaan yang dikirimkan oleh pengguna.

- Latensi: Metrik untuk mengetahui waktu yang dibutuhkan pada saat *interest packet* dikirimkan sampai diterimanya *data packet* oleh pengguna. Satuan latensi yang digunakan adalah mili/second (ms).
- *Path Stretch*: Metrik yang mengukur bentangan jalur digunakan. Sehingga didapatkan rasio antara panjang jalur aktual dan panjang jalur terpendek yang dapat dicapai. Jalur aktual artinya jalur yang diperlukan oleh pengguna menuju penyedia *origin server*, sedangkan jalur terpendek adalah jalur yang diperlukan pengguna untuk mendapatkan konten tanpa menuju ke penyedia *origin server*.

3.1 *Cache Placement*

Cache placement akan mengatur bagaimana sebuah konten diletakkan pada kumpulan *content store* yang tersebar di jaringan. Peletakan konten pada tiap *content store* perlu mempertimbangkan ukuran penyimpanan yang tersedia serta mempertimbangkan konten apa saja yang akan disimpan. Adapun strategi *cache placement* adalah:

- **Leave Copy Everywhere (LCE):** *Content store* akan melakukan *caching* setiap kedatangan konten yang tidak ada di ruang penyimpanan mereka. Keunggulan yang dimiliki adalah memungkinkan penyebaran konten yang paling cepat melalui jaringan. Strategi ini berdampak pada redudansi yang tinggi karena adanya kemungkinan duplikasi konten pada jalur yang dilalui tersebut. Namun, LCE cocok digunakan pada jaringan dengan lalu lintas data yang tinggi.
- **ProbCache:** ProbCache memiliki *header* tambahan berupa TSI (*Time Since Inception*) pada *interest packet* sedangkan *data packet* memiliki header TSI dan TSB (*Time Since Birth*). *Header* TSI dan TSB memiliki sifat yang sama seperti TTL (*Time To Live*) pada datagram IP. Setiap *node* akan menambahkan nilai 1 pada TSI *interest packet*. *Content store* yang memiliki konten akan melampirkan nilai TSI dari *interest packet* ke *data packet*. Setiap *node* juga menambahkan nilai 1 pada

TSB *data packet*. Sehingga selama perjalanan *data packet* menuju penerima, nilai TSI bernilai tetap yang menunjukkan panjang jalur dari *content flow*. Sementara nilai TSB menunjukkan jumlah *hops* yang telah dilalui oleh *data packet* menuju penerima (Psaras, 2012). Probcache memilih *content store* terbaik untuk menyimpan konten menggunakan probabilitas berdasarkan nilai TSI dan TSB. Strategi ini dapat mengurangi *cache redundancy* sehingga konten yang disimpan pada node lebih bervariasi dalam jangka waktu 10 detik.

3.2. *Cache Replacement*

Cache replacement harus ditentukan ketika konten yang tersimpan pada *content store* telah mencapai batas maksimal ruang penyimpanan. *Cache replacement* akan mengatur konten mana saja yang harus diganti dengan yang baru guna meningkatkan peluang *cache hit* dalam keterbatasan ruang penyimpanan. Adapun strategi *cache replacement* sebagai berikut:

- **Least Recently Used (LRU):** LRU akan melacak *interest packet* yang akan dilayani dan kapan waktu melayani mereka. Pada saat kapasitas penyimpanan *content store* penuh, LRU akan mengganti konten yang lama dengan yang baru, maka dipilih salah satu yang paling terakhir tidak diminta oleh pengguna.
- **Least Frequently Used (LFU):** LFU memiliki *counter* yang berkaitan dengan tiap konten pada *content store*. *Counter* akan bertambah ketika konten yang berkaitan diminta oleh *interest packet*. Pada saat kapasitas penyimpanan *content store* penuh, maka strategi ini akan menyisipkan konten baru dengan cara menggantikannya dengan konten yang memiliki nilai *counter* paling sedikit.

4. ICARUS – SIMULATOR ICN

Icarus merupakan simulator *caching* berbasis Python yang digunakan untuk mengevaluasi strategi *In-network Caching* pada arsitektur ICN (Saino, Psaras, & Pavlou, 2014). Simulator Icarus memiliki dua keunggulan, yakni *scalability* dan *extensibility*. *Scalability* yang dimiliki oleh Icarus adalah kehandalannya dalam melakukan simulasi ratusan ribu permintaan per-menit, sehingga mampu menyelesaikan simulasi pengujian yang terdiri dari jutaan permintaan dalam beberapa menit. *Extensibility* berarti menyediakan kemudahan bagi peneliti untuk mengembangkan pengujian lebih luas.

5. METODOLOGI PENGUJIAN

Pengujian dilakukan dengan Icarus berdasarkan alur kerja Icarus, yaitu *Scenario Generation, Experiment Orchestration, Experiment Execution,* dan *Results Collection*. Tahap pertama pengujian adalah *Scenario Generation,* yakni Icarus menyiapkan pengujian berdasarkan berkas konfigurasi dan menyiapkan topologi jaringan yang akan digunakan. Total kombinasi strategi yang akan diujikan berjumlah 4 dan berkas konfigurasi pengujian berjumlah 2 berdasarkan 2 strategi *cache replacement*, yakni LFU dan LRU. Setiap berkas konfigurasi pengujian dengan Icarus hanya dapat menggunakan 1 jenis strategi *cache replacement.*

Selanjutnya pada tahap kedua, yaitu *Experiment Orchestration*, Icarus akan membaca parameter-parameter dari berkas konfigurasi yang telah dibuat yang berisikan parameter dan nilai yang digunakan sesuai dengan Tabel 1. Topologi yang digunakan adalah topologi Tiscali dengan total 240 *node*, di antaranya 36 node merupakan cache router, 44 content source, 36 node receiver, dan sisanya 124 node merupakan router biasa.

Content source merupakan *origin server*, yakni sebagai penyedia konten yang asli. *Cache router* sebagai penyimpan konten yang bersifat sementara dan juga berfungsi untuk meneruskan konten. *Router* biasa hanya berfungsi untuk meneruskan konten ke *node-node* lainnya di dalam jaringan, sedangkan *receiver* merupakan node yang berperan sebagai penerima konten. *Node cache router* pada pengujian merupakan *content store* dalam pengertian struktur data pada ICN. Selanjutnya, istilah *cache router* akan menggantikan peran *content store* pada penelitian ini.

Parameter pengujian nilai α yang digunakan sebesar 0.65 dan 2.0. Nilai α mewakili tingkat popularitas konten di jaringan dengan distribusi *Zipf*. Nilai α 0.65 merepresentasikan *low popularity*, sedangkan nilai α 2.0 sebagai *high popularity*.

Nilai katalog yang digunakan sebesar 100.000. Nilai katalog tersebut mewakili jumlah konten keseluruhan di dalam jaringan.

Sedangkan untuk kapasitas *cache* bernilai 2% atau 4.000 dan 4% atau 8.000 yang tersebar pada 36 *cache router* sehingga tiap *cache router* mampu menyimpan 111 dan 222 konten.

Nilai *warm-up request* disesuaikan dengan jumlah konten yang ada, yaitu 100.000 yang bertujuan untuk mengisi konten pada *cache router* sebelum menjalankan pengujian yang sesungguhnya. Total permintaan yang akan diminta adalah 200.000 dengan asumsi bahwa konten akan diminta 2 kali dari total jumlah konten. Pada nilai *request rate* disesuaikan dengan jumlah *receiver* dalam topologi Tiscali, yakni 36 permintaan per detik.

Tabel 1. Parameter Pengujian

Parameter	Nilai
Topologi	TISCALI
Cache Placement	ProbCache dan LCE
Cache Replacement	LRU dan LFU
Nilai Katalog	100.000
Nilai α	0.65 dan 2.0
Kapasitas Cache	2% dan 4% dari jumlah seluruh konten
Request Rate	36 per detik
Warm-up Request	100.000
Measured Request	200.000

Tahap ketiga adalah *experiment execution*, pada tahap ini melakukan pengujian dan skenario simulasi berdasarkan berkas konfigurasi yang dijalankan pada Icarus. Icarus akan mengukur metrik kinerja yang sudah ditentukan berdasarkan CHR, latensi, dan *path stretch.*

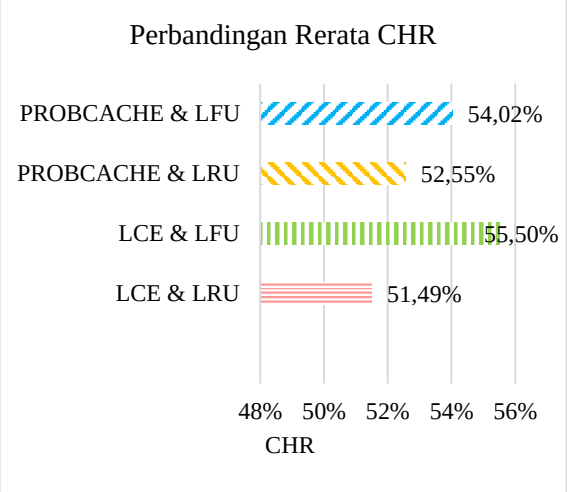
Setelah pengujian berhasil dilakukan, maka tahap terakhir adalah *results collection*. Tahap ini akan menghasilkan dua berkas hasil pengujian yang masih berformat data *pickle* sehingga perlu dirubah menjadi format data *text* untuk mempermudah pengolahan data hasil pengujian. Berkas hasil pengujian berisikan nilai yang didapatkan berdasarkan metrik kinerja CHR, latensi dan *path stretch* dari tiap kombinasi strategi, yakni ProbCache & LFU, ProbCache & LRU, LCE & LFU, dan LCE & LRU.

6. HASIL DAN PEMBAHASAN

Setelah pengujian dilakukan maka tahap selanjutnya adalah evaluasi terhadap kinerja *In-network Caching*. Hasil pengujian berisikan 2 hasil kombinasi strategi pada tiap berkas konfigurasi sesuai dengan kombinasi strategi *cache placement* ProbCache dan LCE dengan *cache replacement* LFU dan LRU. Selanjutnya, evaluasi dan pembahasan yang dilakukan dengan cara membandingkan hasil pengujian berdasarkan metrik CHR, latensi, dan *path stretch.*

6.1 *Cache Hit Ratio*

Nilai CHR yang didapatkan merupakan rerata perbandingan antara jumlah permintaan yang berhasil dilayani oleh *cache router* atau disebut dengan *cache hit* dengan total permintaan yang dikirimkan oleh pengguna. Nilai CHR yang cenderung bernilai 100%, maka artinya jumlah permintaan yang dikirim oleh pengguna juga cenderung lebih sering mengalami *cache hit*.



Gambar 3. Perbandingan Rerata CHR

Gambar 3 merupakan grafik perbandingan hasil rerata nilai CHR yang didapatkan dari seluruh pengujian. Kombinasi strategi di *in-network caching* menunjukkan dampak pada besarnya nilai CHR yang didapatkan. Hasil rerata CHR menunjukkan LCE & LRU mendapatkan nilai CHR terendah dibandingkan strategi lainnya dengan nilai 51,49%. Sedangkan nilai CHR tertinggi didapatkan oleh LCE & LFU dengan nilai 55,50%. Kedua strategi, LCE & LRU dan LCE % LFU sama-sama menggunakan strategi *cache placement* LCE, namun perolehan CHR selisih sebesar 4,01% dapat terjadi karena perbedaan strategi *cache replacement* yang digunakan.

Nilai CHR yang tinggi diperoleh LCE

mendukung teori mengenai cara kerja LCE, yakni *cache router* akan melakukan penyimpanan konten setiap ada konten baru yang belum ada di *cache router* tersebut. Adapun keunggulannya dari LCE adalah kecepatan penyebaran konten yang tersebar di dalam jaringan.

LFU yang merupakan *cache replacement* mampu mendukung kinerja LCE dengan baik dibandingkan dengan LRU (selisih 4,01% lebih tinggi), meski pun tingkat komputasi pencarian dan penggantian lebih tinggi dibandingkan dengan LRU. LFU memiliki *counter* yang memantau setiap konten yang disimpan pada *cache router*, sehingga ketika konten yang akan diganti dengan konten baru dapat dipastikan merupakan konten yang paling jarang diminta berdasarkan nilai *counter* yang bernilai kecil. Hal ini berdampak pada nilai CHR yang tinggi karena *cache router* hanya menyimpan konten-konten terbaru dan yang paling sering diminta sehingga peluang *cache hit* menjadi besar.

6.2 Latensi

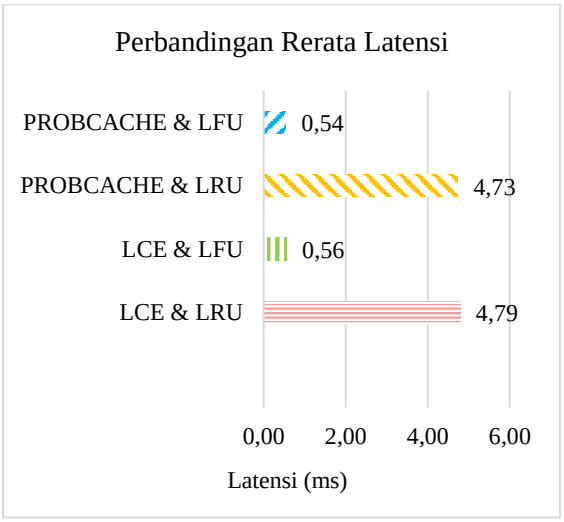
Latensi merupakan waktu yang diperlukan bagi *receiver* mulai dari permintaan dikirimkan sampai menerima konten yang diminta. Nilai latensi yang rendah artinya penerima mendapatkan konten yang diminta dengan segera, sedangkan nilai latensi yang tinggi artinya proses pengiriman konten yang diminta ke *receiver* lebih lambat.

Gambar 4 merupakan grafik perbandingan rerata nilai latensi yang didapatkan dari seluruh pengujian. Hasil yang didapatkan menunjukkan bahwa latensi terendah diraih oleh ProbCache & LFU dengan nilai rerata 0,54 ms. Sedangkan nilai latensi tertinggi diraih oleh LCE & LRU. Padahal pada strategi LCE & LRU mendapatkan nilai latensi rendah senilai 0,56 ms (selisih 0,02 ms dengan ProbCache & LFU). Nilai latensi yang tinggi pada LCE & LRU disebabkan karena cara kerja LRU yang kurang sesuai dengan LCE sehingga tidak memaksimalkan kinerja strategi tersebut.

LCE menyimpan konten-konten baru ke seluruh *cache router* dengan segera ketika kapasitas *cache* penuh, namun LRU hanya memilih konten mana yang terakhir tidak digunakan dan tidak mempertimbangkan frekuensi konten tersebut diminta. Sehingga ada kemungkinan konten yang diminta tidak dimiliki oleh *cache router*, maka terjadi *cache miss* dan memperbesar PIT karena permintaan akan diteruskan ke *cache router* yang lain sampai menemukan konten yang diminta. Selanjutnya, PIT yang terjadi akan memperbesar nilai latensi karena *cache miss*.

ProbCache & LFU mendapatkan nilai latensi terendah pada kedua kondisi *popularity*, yaitu *low* dan *high popularity*. Strategi ProbCache melakukan penyimpanan konten dengan cara perhitungan probabilitas, sehingga dapat mengurangi *cache redundancy* karena konten yang disimpan pada *cache router* lebih bervariasi dalam jangka 10 detik.

Berdasarkan gambar 4 menunjukkan bahwa, pada sisi *cache replacement*, LFU mampu menurunkan nilai latensi dibandingkan dengan LRU sebesar 4,19 ms. Selisih nilai latensi antara LFU dan LRU ini menunjukkan bahwa ProbCache mampu bekerja dengan optimal dengan LFU dibandingkan dengan LRU. Meski pun LRU yang memiliki tingkat komputasi pencarian dan penggantian lebih sederhana dibandingkan dengan LFU, namun tidak menjamin nilai latensi yang didapatkan lebih rendah.



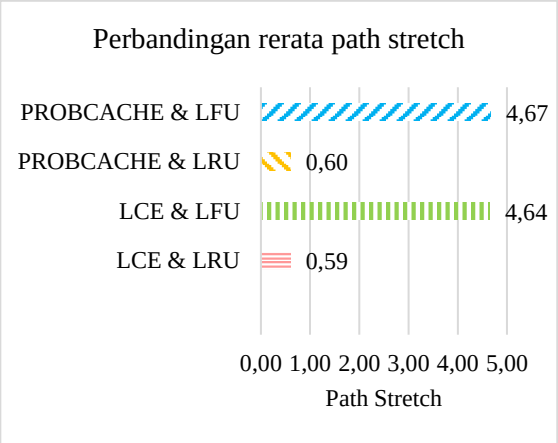
Gambar 4. Perbandingan Rerata Latensi

6.3 *Path Stretch*

Path stretch merupakan metrik yang mengukur rasio bentangan jalur yang aktual dengan jalur terpendek yang dapat dicapai.

Gambar 5 menunjukkan perbandingan rerata nilai rasio *path stretch* dari seluruh strategi *in-network caching* yang telah diujikan. Gambar 5 menunjukkan bahwa LCE & LRU memiliki nilai rerata *path stretch* terendah dibandingkan strategi lainnya. Nilai rasio *path stretch* yang rendah menunjukkan bahwa strategi LCE &

LRU cenderung tidak mampu memotong jalur aktual dari lokasi original konten sehingga *client* tidak mendapatkan konten yang diminta dengan segera. Maka, sesuai dengan hasil yang didapatkan pada subbab 6.2 bahwa nilai latensi tertinggi didapatkan dari strategi LCE & LRU. Sebab ketika nilai rasio *path stretch* rendah, maka paket *interest* perlu mencari konten lebih jauh yang mengakibatkan nilai latensi yang tinggi.



Gambar 5. Perbandingan Rerata *Path Stretch*

Berdasarkan Gambar 5, strategi ProbCache & LFU mendapatkan rerata nilai *path stretch* tertinggi di antara strategi lainnya. Nilai rasio *path stretch* yang tinggi berarti strategi tersebut mampu memotong jalur aktual dari lokasi original konten dengan strategi *In-Network Caching* yang efisien, sehingga berdampak pada nilai latensi yang rendah karena konten yang diminta menjadi lebih dekat di sisi *client*. Strategi ProbCache & LFU memiliki nilai rasio *path stretch* tertinggi di antara strategi lainnya. Hal ini sesuai dengan hasil pengujian latensi yang menunjukkan bahwa ProbCache & LFU memiliki nilai latensi terendah.

7. KESIMPULAN

Berdasarkan hasil evaluasi dan perbandingan yang diperoleh, strategi in-network caching yang memiliki kinerja terbaik pada kondisi *low* dan *high popularity* adalah, ProbCache & LFU yang memiliki latensi terendah dan path stretch tertinggi di antara seluruh strategi in-network caching lainnya yang telah diujikan, meskipun CHR tertinggi diperoleh LCE & LFU (selisih 0,49% lebih rendah dibandingkan LCE & LFU).

Adapun kesimpulan lainnya, yaitu semakin besar nilai α dan kapasitas *cache* akan berdampak pada CHR yang semakin tinggi. Namun nilai CHR yang tinggi tidak menjamin latensi yang lebih rendah dan *path stretch* yang tinggi. Sedangkan nilai rasio *path stretch* yang tinggi berdampak pada nilai latensi yang rendah. Perpaduan antara strategi *cache placement* dan *cache replacement* yang sesuai juga berdampak pada peningkatan CHR, penurunan latensi dan peningkatan rasio *path stretch*. Sehingga strategi yang terbaik tetap memiliki kinerja yang unggul, baik di kondisi *low popularity* mau pun *high popularity*.

8. DAFTAR PUSTAKA

Yusung Kim, Ikjun Yeom, 2013. Performance Analysis of In-Network Caching for Content-Centric Networking. Korea, Sung Kyun Kwan University.

Jacobson, V., 2009. A Description of Content Centric Networking (CCN). German: Parc Company.

Rossi Dario, Giuseppe Rossini, 2011. Caching Performance of Content Centric Networks Under Multi-path Routing. Paris: Telecom Paris Tech.

Wang, Yonggong et al, 2013. Optimal Cache Allocation for Content-Centric Networking. Proceedings - International Conference on Network Protocols.

Saino Lorenzo, Psaras, I., & Pavlou, George, 2014. Icarus: a Caching Simulator for Information Centric Networking (ICN).

Maulana, Muhammad Rizka, Basuki, A., & Amron. Kasyful, 2018. Analisis Kinerja On-Path Caching dan Off-Path Caching pada Information-Centric Networking.

Bernardini, C., Silverston, T., & Festor Olivier. 2015. A Comparison of Caching Strategies for Content Centric Networking.

Xylomenos, George et. al., 2013. A Survey of Information-Centric Networking Research.

Yaqub, Muhammad A et. al., 2016. An Overview, Applications and Research Challenges.

Guoqiang Zhang, Yang Li, & Tao Lin, 2013. Caching in Information Centric Networking: A Survey.

Pfender Jakob, Alvin Valera, & Winston K.G.

Seah, 2018. Performance Comparison of Caching Strategies for Information-Centric IoT.

Ioannis Psaras, Wei Koong Chai, and George Pavlou, 2012. Probabilistic InNetwork Caching for Information-Centric Networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking.

Bernardini Cesar, Thomas Silverston, Oliver Festor, 2015. A Comparison of Caching Strategies for Content Centric Networking. IEEE Global Communication Conference.

Breslau, L & Cao, Pei & Fan, Li & Phillips, Graham & Shenker, S. (1999). Web Caching and Zipf-Like Distributions: Evidence and Implications. Proceedings - IEEE INFOCOM. 1. 126 - 134 vol.1. 10.1109/INFCOM.1999.749260.