OPTIMASI *REPLACEMENT STRATEGY* UNTUK METODE *PROGRESSIVE CACHE* PADA *CONTENT CENTRIC NETWORKING* (CCN)

Arief Prasetyo[1)], Royyana Muslim Ijtihadie[2)], dan Tohari Ahmad[3)]
[1, 2)]Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia
e-mail: arief.prasetyo@polinema.ac.id[1)], roy@if.its.ac.id[2)], tohari@if.its.ac.id[3)]

ABSTRAK

Teknologi internet yang berkembang pesat, menyebabkan meningkatnya jumlah konten multimedia pada lalu lintas jaringan data. Content Centric Network (CCN) dengan fitur yang dimilikinya yaitu content distribution, multicast, in-network cache, mobility dan delay-tolerant networking, dikembangkan untuk memenuhi tantangan yang muncul pada model komunikasi host-to-host (teknologi internet saat ini), yaitu mendistribusikan konten-konten multimedia dengan lebih efisien. Performa CCN terbukti dapat ditingkatkan dengan menerapkan metode progressive cache pada strategi penyimpanan cache. Metode Progressive Cache menggunakan prinsip kerja meminimalisir data one-timer, mempertimbangkan popularitas data dan frekuensi request data. Pada penelitian ini, metode Progressive Cache dioptimasi dengan merubah strategi cache replacement yang digunakan untuk lebih meningkatkan performa jaringan CCN yang dapat diterapkan pada segala jenis topologi jaringan. Optimasi dilakukan dengan menerapkan metode LRUd (LRU with distance), yang memperhitungkan recency serta jarak sumber data terhadap caching node pada strategi cache replacement dari Progressive Cache. Metode LRUd diterapkan pada seluruh node jaringan CCN. Hasil yang didapatkan dengan menerapkan metode LRUd dengan Progressive Cache pada jaringan CCN dapat meningkatkan cache hit-rate hingga 0.416%, menurunkan hop distance hingga 1,145% dan meningkatkan jumlah download hingga 0,722%.

***Kata Kunci**: cache replacement strategy, progressive cache, CCN, LRU, LRUd*

ABSTRACT

Internet technology are rapidly evolving, causing the increasing number of multimedia content on the network data traffic. Content Centric Network (CCN) and its features: content distribution, multicast, in-network cache, mobility and delay-tolerant networking, has been developed to meet the challenges that arise in the host-to-host communication model (today's internet technology), that is to distribute multimedia content more efficiently. CCN performance can be improved by implementing progressive cache method. Progressive Cache principles are avoiding one-timer data in cache, considering the popularity of the data and considering the frequency of the data request. In this study, the Progressive Cache method is optimized by changing the replacement strategy to further improve network performance CCN that can be applied to any type of network topology. Optimization is done by applying the method LRUd (LRU with distance), which take into account the recency and distance data sources to caching nodes on the replacement strategy of Progressive Cache. LRUd method can be applied to the all CCN's node. The results obtained by applying the LRUd method with Progressive Cache on CCN network can improve the cache hit-rate up to 0416%, lower hop-distance up to 1.145% and increase the number total downloads of up to 0.722%.

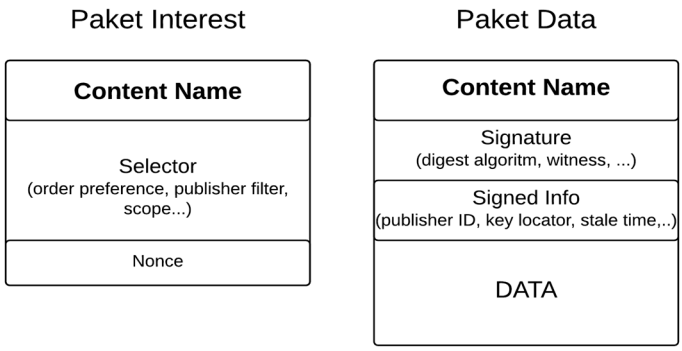
***Keywords**: cache replacement strategy, progressive cache, CCN, LRU, LRUd*

I. PENDAHULUAN

TEKNOLOGI INTERNET saat ini sudah berkembang sangat pesat dibandingkan dengan awal diciptakannya teknologi jaringan di tahun 60-70an. Pada awal diciptakannya, teknologi jaringan memiliki model komunikasi *host-to-host* dengan tujuan berbagi sumber daya [1]. Seiring dengan murahnya harga perangkat komputer, dan terjangkaunya harga *bandwidth broadband*, maka terjadi pergeseran fungsi Internet menjadi wadah ledakan konten dan sarana berbagi informasi baru yang dihasilkan oleh pengguna Internet. Pada tahun 2008, konten yang dihasilkan di Internet mencapai jumlah 500 exabyte [2].

Dengan komunikasi TCP/IP yang berorientasikan *host*, dan trafik yang semakin besar, model layanan berbasis TCP/IP ini menunjukkan keterbatasannya. Beberapa permasalahan yang timbul dari tidak sesuainya model-model internet adalah [3]:

- *availability* → akses terhadap konten yang cepat dan handal membutuhkan mekanisme spesifik terhadap aplikasi tertentu seperti CDN dan jaringan P2P, yang juga membutuhkan bandwidth yang sangat besar.
- *security* → tingkat kepercayaan pada konten sering kali tidak tepat, mengandalkan lokasi konten dan informasi koneksi yang tidak dapat dipercaya.
- *location-dependence* → memetakan konten ke lokasi host membuat semakin rumitnya konfigurasi dan implementasi servis jaringan.



Gambar 1. Paket Interest dan Paket Data pada jaringan CCN[1]

Saat ini *Content Centric Network* (CCN) dikembangkan sebagai *overlay network* untuk mengatasi permasalahan-permasalahan tersebut. CCN bertujuan mendesain ulang arsitektur internet dengan *named-data* sebagai elemen utama pada paradigma baru komunikasi. Pada CCN, data dibentuk dalam serangkaian potongan-potongan (*data chunk*) yang diidentifikasikan dengan menggunakan penamaan secara hirarki dan selanjutnya dapat diminta oleh *client* menggunakan paket *Interest* untuk setiap potongan data yang diinginkan. Fitur yang dimiliki CCN antara lain adalah *content distribution, multicast, mobility* dan *delay-tolerant networking* [4]. Semua fungsi tersebut bergantung pada *cache*/replikasi data yang tepat di setiap *node / router*, yang mana sudah merupakan karakteristik dasar dari CCN di mana setiap router melakukan *cache in-network*.

Beberapa metode *caching* yang umum, telah diteliti implementasi dan hasil kinerjanya pada jaringan CCN [10]. Pada penelitian [4], diketahui bahwa metode *Progressive Caching Policy* (PCP) yang diterapkan pada *node-node* jaringan CCN dapat meningkatkan kinerja CCN. Metode PCP menerapkan strategi *replacement* yang berbeda untuk *intermediate node* dan *edge-node*. Hal itu mengindikasikan bahwa metode tersebut lebih tepat untuk diterapkan pada topologi-topologi jaringan yang statis dan memiliki hirarki antara sumber data hingga ke *client*. Karena CCN dikembangkan dengan salah satu alasan untuk mempermudah mobilitas dan skalabilitas jaringan, maka perlu dilakukan modifikasi pada strategi *cache replacement* metode *Progressive Cache Policy*, dengan tujuan agar dapat diterapkan pada jaringan CCN dengan topologi yang dinamis.

Penelitian ini mengusulkan metode LRUd (LRU *with distance*), strategi *replacement* yang memperhitungkan *recency* seperti pada LRU biasa, dan *distance* (jarak antara *node* yang meminta data dengan *node* sumber data), untuk mengganti strategi *replacement* pada metode asli *Progressive Cache Policy*, dan mempelajari kinerjanya pada jaringan CCN dalam simulasi. Strategi *replacement* yang digunakan tidak membedakan *edge node* dan *intermediate node*, sehingga dapat diterapkan pada segala topologi jaringan CCN serta tetap meningkatkan performa *cache* jaringan CCN.

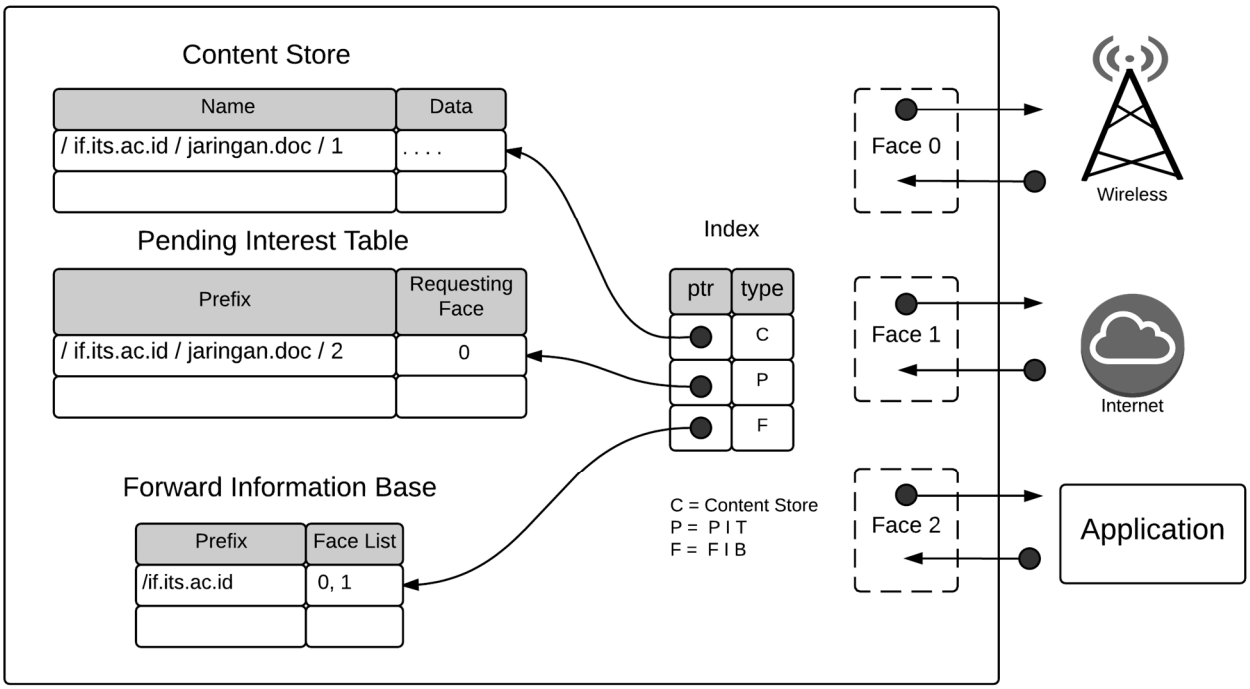
II. PENELITIAN TERKAIT

A. Model Content Centric Networking (CCN)

CCN merupakan arsitektur jaringan baru yang merubah paradigma komunikasi berbasiskan lokasi *host* menjadi komunikasi yang berbasiskan konten[1]. Inti dari jaringan CCN adalah pemberian nama konten terhadap data yang dikomunikasikan. Hal ini berbeda dengan komunikasi berbasis lokasi atau komunikasi *host-to-host*, dimana identifikasi data berdasarkan pengalamatan *host* peminta data dan *host* pengirim data. Pada CCN data diidentifikasi berdasarkan penamaan sesuai konten yang dimilikinya. Data dipartisi menjadi potongan-potongan data yang disebut dengan *chunk*, dan diberi nama secara hirarki. Sebagai contoh, sebuah *chunk* bernama `/if.its.ac.id/jaringan.doc/1` bisa diidentifikasi sebagai *chunk* pertama dari file `jaringan.doc` yang dimiliki oleh host dengan nama `if.its.ac.id`.

Pada model jaringan CCN, *node* yang menghasilkan *content* disebut dengan *publisher*, dan *node* yang meminta *content* disebut dengan konsumen. Tipe paket pada jaringan CCN ada dua tipe, paket *Interest* dan paket *Data,* masing-masing paket memiliki *identifier* berupa nama konten seperti yang dapat dilihat pada Gambar 1.

Node pada CCN bekerja hampir sama dengan *node* pada komunikasi IP. Jika sebuah paket tiba pada sebuah *face* (*face* adalah sebutan pengganti *interface* pada jaringan CCN), pencarian berdasarkan kecocokan yang terpanjang dilakukan terhadap nama yang terkandung di dalam paket tersebut. Lalu proses berikutnya dikerjakan berdasarkan hasil pencarian sebelumnya. Skema proses *forwarding* digambarkan pada Gambar 2. Mesin *forwarding* yang ada pada setiap *node* jaringan CCN memiliki 3 komponen utama: FIB (*forwarding interest base*), *Content Store* (memori *buffer*) dan PIT (*Pending Interest Table*).



Gambar 2. Skema proses forwarding pada node CCN [1]

FIB digunakan untuk melakukan proses *forwarding* paket *Interest* menuju ke lokasi yang berpotensi memiliki sumber data yang diminta. FIB pada CCN bisa mencatat daftar lebih dari satu *face* untuk masing-masing sumber data, berbeda dengan FIB pada jaringan IP yang hanya mencatat satu *interface* untuk setiap *outgoing interface* menuju jaringan IP tertentu.

Content Store merupakan memori *buffer* untuk menyimpan data seperti pada *router* jaringan IP, tetapi memiliki metode replacement yang berbeda. Paket-paket IP merupakan paket yang khusus diperuntukkan komunikasi *point-to-point*, sehingga setelah paket tersebut berhasil di-*forward*, maka paket tersebut akan segera dihapus. Pada CCN paket data bersifat *idempotent, self-identifying* dan *self-authenticating*, sehingga berpotensi berguna bagi konsumen lain yang akan meminta data dengan konten yang sama. *Node* pada CCN akan menyimpan selama mungkin setiap data yang ada, untuk meminimalisir penggunaan *bandwidth upstream* dan *download latency*.

PIT adalah tabel yang terdapat pada *node* CCN, yang mencatat setiap *face* tempat datangnya paket *Interest*. Sehingga apabila ada paket Data yang sesuai datang, paket Data tersebut bisa dikirimkan ke seluruh *face* yang meminta data yang bersangkutan. Setiap paket Data dikirimkan, entri paket *Interest* terhadap data tersebut yang tercatat pada PIT akan dihapus.

Setiap *node* CCN mencatat dari *face* mana setiap *interest* yang datang, ke dalam PIT-nya, di mana setiap entri berisi nama dari paket *Interest* dan daftar *face* dari mana paket *Interest* itu diterima. Saat ada beberapa *Interest* untuk data yg sama diterima dari *downstream*-nya, hanya 1 *interest* yang di kirim ke *upstream* menuju sumber data. Paket *interest* tersebut diteruskan / *forward* ke *node* yang kemungkinan memiliki data yang diinginkan, dengan melihat daftar *face* pada FIB. Entri pada FIB dihasilkan oleh protokol *name-based routing*. Saat paket *Interest* sampai ke *node* yg memiliki data yang dimaksud, paket Data akan dikirimkan kembali, yang berisi nama dan konten data, serta penanda dari produser data (*producer's key*). Paket Data melacak kembali jalur yg ditempuh oleh paket *Interest* dengan mengikuti entri PIT setiap *node / router* yang dilewatinya, hingga sampai ke konsumen. Setiap paket Data melewati *node*, maka entri yang bersangkutan pada PIT *node* tersebut dihapuskan bersamaan dengan proses *forwarding* paket Data dan menyimpan *cache* Data pada *Content Store*. Tidak ada alamat *host* / *interface* di masing-masing paket *Interest* dan Data. Paket *Interest* dipilihkan rute berdasarkan nama yang ada dalam paketnya, dan paket Data dikirimkan berdasarkan informasi status yang dibentuk saat paket Interest berpindah *node*.

Seperti disebutkan sebelumnya, paket Data CCN tidak diidentifikasi dari lokasi *node* asal dan lokasi *node* tujuan, sehingga *router* dapat menyimpan dalam *cache* untuk memenuhi *potential future request*. Hal ini berarti CCN secara otomatis mendukung banyak fungsi tanpa infrastruktrur extra, termasuk distribusi konten (banyak *user* meminta data yang sama pada waktu yang berbeda), *multicast* (banyak *user* meminta data yang sama pada

waktu yang sama), mobilitas (*user* meminta data dari lokasi berbeda) dan jaringan *delay-tolerant* (user memiliki koneksi *intermittent*) [1].

B. Cache dan Progressive Caching Policy

Setiap *node / router* pada jaringan CCN, memiliki *Content Store*, yang pada dasarnya merupakan *buffer memory* pada router, atau dapat disebut juga dengan *cache*. Metode *cache* mulai diperkenalkan pada dunia jaringan dan menunjukkan pentingnya keberadaannya seiring dengan berkembangnya Internet dan Web serta konten-kontennya. Fitur yang dimiliki oleh *cache* [7] adalah *cache* dapat mengurangi penggunaan *bandwidth*, mengurangi *delay* dari sisi *user* serta mampu mengurangi *load* pada server asli.

Cara kerja proses *cache* secara garis besar dapat dibedakan menjadi 2 proses: proses pemilihan data yang akan disimpan pada cache (*decision strategy*), serta proses pemilihan data yang akan dihapus dari cache apabila memori sudah penuh terisi (*replacement strategy*). Saat pertama kali sistem *cache* dijalankan, memori *cache* masih kosong. Setiap data yang datang pada *caching node* akan melalui proses pertama, untuk menentukan apakah data tersebut disimpan dalam memori *cache* atau tidak. Pada saat memori yang digunakan untuk menyimpan data pada *cache* telah penuh, maka *cache* tidak dapat menyimpan data baru. Metode *replacement* digunakan untuk menghapus data yang ada pada *cache* yang penuh, untuk digantikan dengan data baru. Banyak metode *replacement* yang diteliti sebelumnya. Faktor yg biasa diperhitungkan untuk proses *replacement* pada *cache* [8]:

- *recency*: waktu terakhir referensi ke objek
- frekuensi: jumlah *request* ke sebuah objek
- *size*: ukuran dari objek dalam *cache*
- *cost of fetching the object*: perhitungan pengambilan objek langsung dari server asli (seperti jarak dalam jumlah hop)
- *modification time*: waktu dimodifikasi terakhir sebuah objek
- *expiration time*: waktu dimana objek sudah kadaluarsa dan bisa diganti langsung

Progressive Cache Policy (PCP) yang diusulkan pada [4], juga terdiri dari *cache decision strategy* dan *replacement strategy*. Algoritma *decision strategy* yang digunakan pada PCP memiliki kemiripan dengan *decision strategy Leave Copy Down* (LCD). Algoritma I menunjukkan algoritma *decision strategy* yang digunakan PCP. Algoritma II dan algoritma III menunjukkan *replacement strategy* yang digunakan PCP, dimana ada perbedaan strategi yang diterapkan pada *Edge Node* dan *Intermediate Node*. Hal ini menyebabkan metode PCP lebih tepat diterapkan pada topologi yang tetap dan bersifat hirarki.

Replacement Strategy PCP yang diterapkan pada *edge node* mengelompokkan data chunk yang disimpan pada cache berdasarkan frekuensi dan diurutkan berdasarkan *recency* (yang terakhir diakses berada pada urutan terdepan). Setiap kali memori *cache* penuh dan harus menghapus data pada cache agar data baru bisa disimpan, maka data dengan frekuensi akses paling sedikit serta yang waktu aksesnya paling lama yang dipilih untuk dihapus. Sedangkan *replacement strategy* PCP pada *intermediate node*, menggunakan faktor *recency* dan *random* untuk memilih data *chunk* mana yang akan dihapus dari memori *cache* apabila cache telah penuh dan ingin menyimpan data *chunk* yang baru pada *cache*.

Untuk mengukur performa dari strategi yang diterapkan suatu sistem *cache*, menurut [7] ada beberapa metrik yang dipergunakan, antara lain adalah:

- *hit-rate* → metrik yang paling sering digunakan, metrik ini digunakan untuk menentukan seberapa banyak persentase dari hit lokal.
- *byte-hit rate* → strategi yang cenderung menghilangkan objek berukuran besar akan menaikkan nilai *hit-rate* dan mengurangi *byte-hit rate*. Metrik kinerja ini menarik bagi ISP di mana ISP cenderung mengurangi volume *download* langsung dari internet.
- *delay saving ratio* → metrik ini juga tepat digunakan dalam sistem interaktif di mana pengukuran dilakukan dengan harapan mengurangi waktu *download*.

Algoritma I. *Decision Strategy* pada *Progressive Cache Policy* [4]

	Saat menerima paket data *p* pada *node n*,
1	jika *n* adalah *downstream node* paling dekat dengan *cache hitting node*, maka *decision = true*;
2	jika n adalah *intermediate node* dan $d_p \geq \theta_1$,
3	*decision = true*;
4	jika n adalah *edge node* dan $r_p \geq \theta_2$, maka *decision* = true;
	jika tidak, *decision = false,*

Algoritma II. *Replacement Strategy Progressive Cache Policy* untuk *Edge Node* [4]

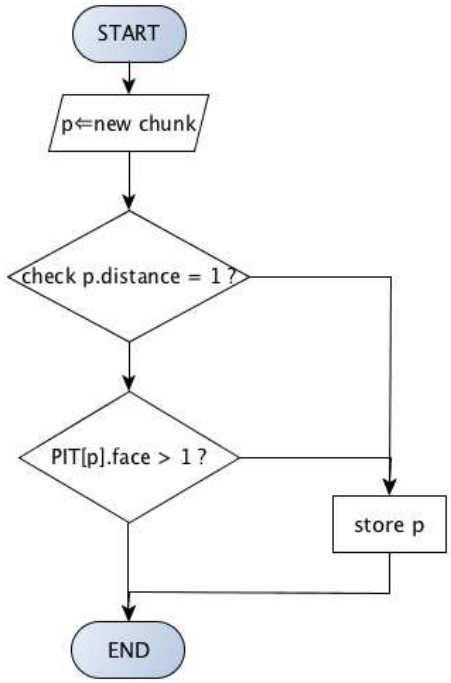
Cache dikelola dalam beberapa antrian: Q[K]
Memberi nilai awal L ← 0
Terjadi *request* terhadap *chunk p*
c_p = jumlah hop untuk mengambil *p*
if *p* sudah ada pada *cache*
    hapus *p* dari antrian sekarang Q[Hp];
    masukkan *p* ke bagian belakang dari Q[H'$_p$], dimana H'$_p$ = L + c_p
if *p* tidak ada pada *cache*
    if diperlukan penghapusan item:
        L ←arg **min**Q[k] ≠ Ø
                $k \epsilon K$
        hapus data chunk q dari bagian depan antrian Q[L],
        contohnya antrian dengan H_q = L.
    masukkan p ke bagian belakang Q[H_p], dimana H_p= L+c_p

Algoritma III. *Replacement Strategy Progressive Cache Policy* untuk *Intermediate Node* [4]

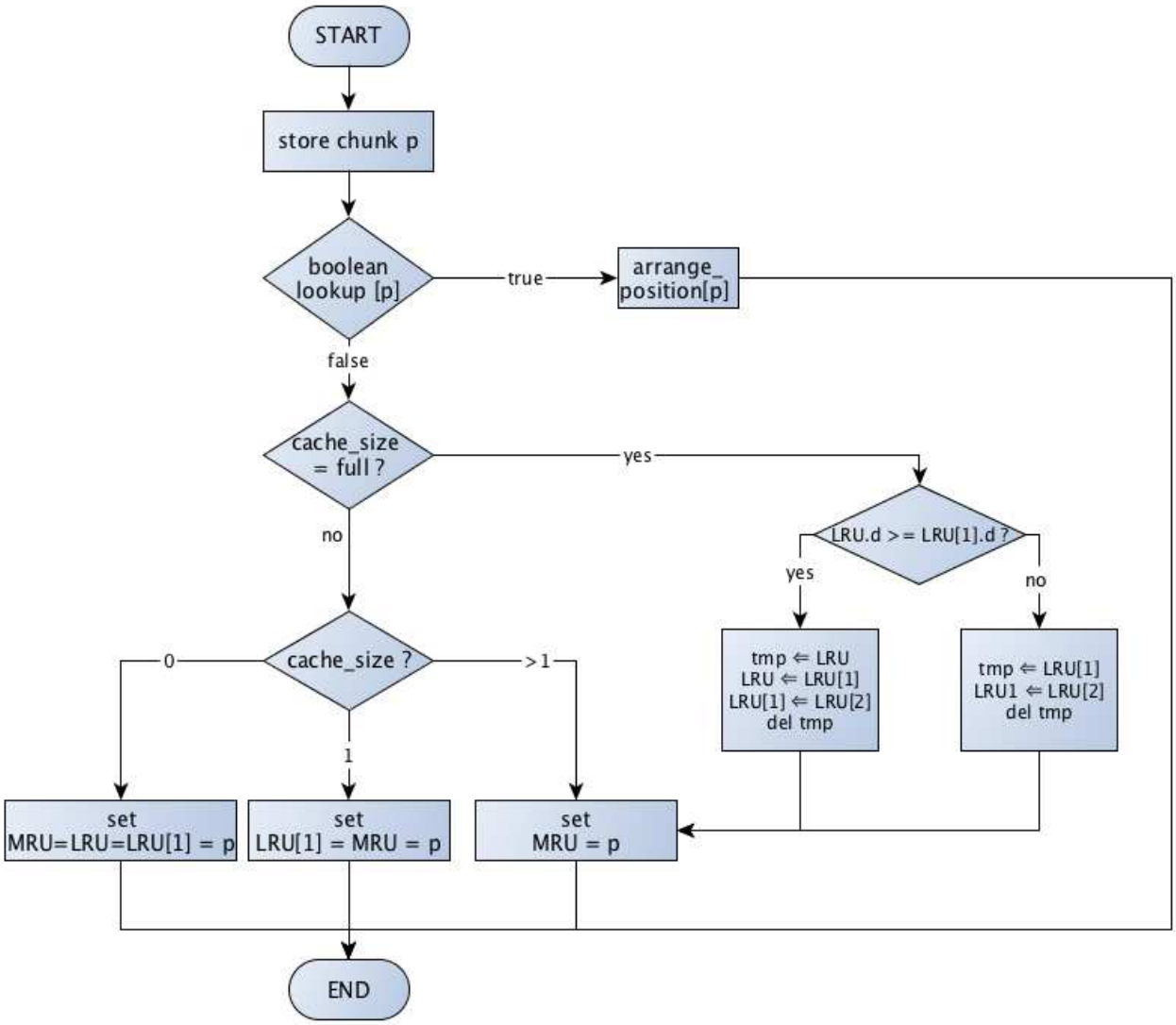
1: Terjadi *request* terhadap *chunk p*
2: if *p* sudah ada pada *cache*
3: *f*: *interface* datangnya *request*
4: if $f \in S_p^1$,
5: set $H_p \leftarrow H_p - c_p$, **if** $f \notin S_p^2$; $S_p^2 = S_p^2 \cup \{f\}$; ;
6: else if $f \notin S_p^1$;
7: $H_p \leftarrow H_p + c_p$; $S_p^1 = S_p^1 \cup \{f\}$;
8: if *p* tidak ada pada *cache*
9: **if** diperlukan penghapusan item:
10: hapus data *chunk u*, s.t., u = arg **min** H_u;
11: pilih secara *random* item *q*; set $H_q \leftarrow [H_q - c_q]+$;
12: masukkan p ke dalam *cache*:
13: set $H_p \leftarrow d_p$ x c_p ; inisialisasi S_p^1; $S_p^2 = \emptyset$

III. METODE YANG DIUSULKAN

Pada penelitian ini kami mengusulkan metode *caching* baru yang dimodifikasi dari PCP dengan tujuan untuk mengoptimalkan kinerja jaringan CCN. *Decision strategy* yang kami gunakan masih bersifat progressif seperti pada PCP, yaitu semakin sering suatu data *chunk* diminta oleh *node* tertentu, *caching node* yang menyimpan data *chunk* tersebut semakin maju mendekati *requesting node*. Diagram alir dari *decision strategy* tersebut dapat dilihat pada gambar 3.



Gambar 3. Diagram Alir *decision strategy Progressive Cache*



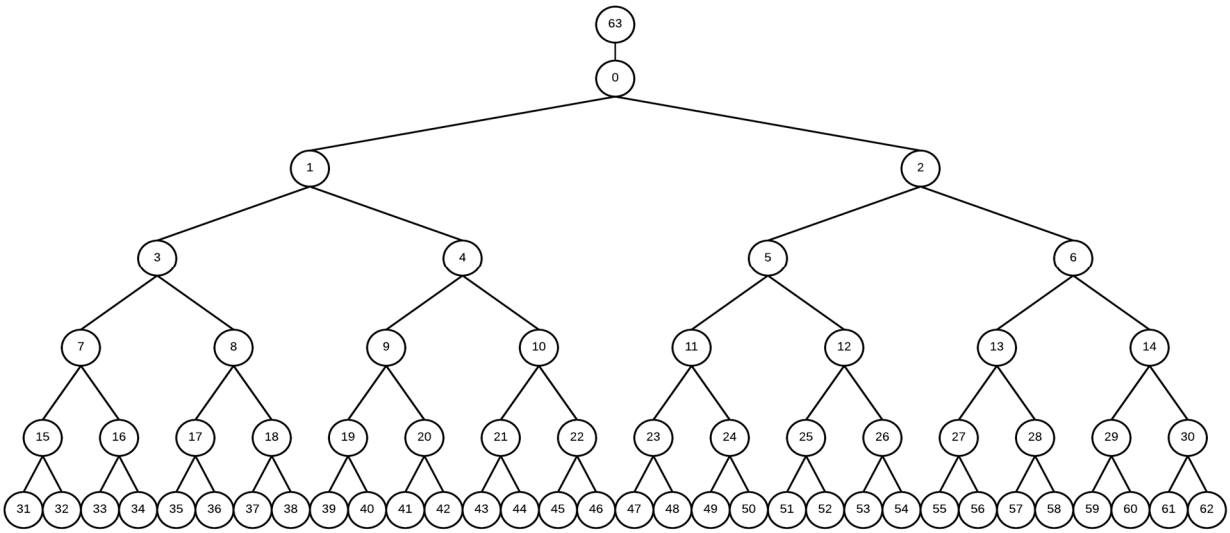
Gambar 4. Diagram Alir *replacement strategy* LRUd (*LRU with distance*)

Algoritma decision strategy PCP yang kami implementasikan pada penelitian ini adalah sebagai berikut, saat sebuah *node* menerima paket data *p* pada *node n*,

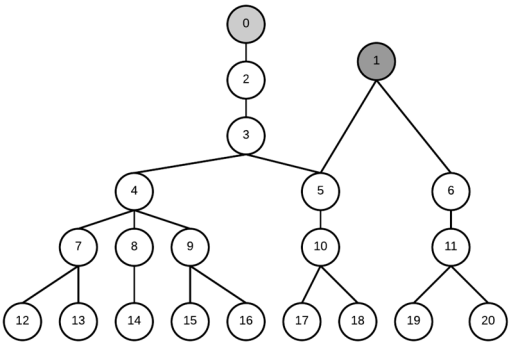
1. jika n adalah *downstream node* paling dekat dengan *cache hitting node (repository)*, maka *decision = true*;
2. jika n adalah *intermediate node* dan memiliki *face* pada PIT > 1, *decision = true*;
3. jika tidak, *decision = false*

Untuk *replacement strategy* yang kami gunakan sebagai pengganti *replacement strategy* orisinil metode PCP, kami menggunakan modifikasi dari LRU (*Least Recent Used*), yang kami sebut LRU-*distance* (LRUd). Berdasarkan penelitian [6], LRU merupakan *replacement strategy* yang kinerjanya baik dibandingkan *replacement strategy* umum lain (UNIF, LFU, MRU, MFU) yang diterapkan pada jaringan CCN. Pada [10] dinyatakan bahwa LRU dipertimbangkan sebagai batas atas kompleksitas strategi yang masih dapat diterapkan pada jaringan CCN. *LRU-distance* merupakan modifikasi dari metode LRU biasa dengan menambahkan parameter *distance* pada pemilihan entri yang akan dihapus dari memori *cache*. *Distance* yang dimaksud adalah jarak antara *caching node* dengan *repository* asal entri yang ada dalam *cache*. LRUd membandingkan antara dua entri terakhir yang diakses dalam *cache*, mana yang sumber *repository*-nya paling jauh. Yang *repository* sumbernya lebih jauh yang akan dipertahankan di dalam *cache*. Untuk lebih jelasnya, diagram alir dari LRUd dapat dilihat pada gambar 4.

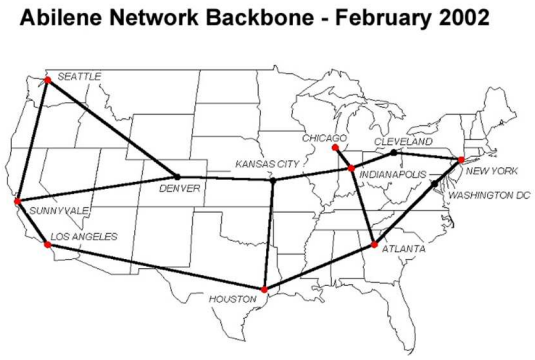
Pada LRUd tidak membedakan *replacement stratey* untuk *edge node* maupun *intermediate node* seperti pada metode orisinil PCP. Dengan demikian, PCP dengan LRUd dapat diimplementasikan pada topologi yang dinamis dan tidak hirarki. Metode yang kami usulkan ini kami terapkan dan analisis pada beberapa topologi, seperti yang dibahas pada bab berikutnya tentang uji coba.



(a)



(b)



(c)

Gambar 5. (a) topologi tiyok; (b) topologi tiyok; (c) topologi abilene[11]

IV. UJI COBA

A. Skenario Uji Coba

Uji coba penelitian ini dilakukan dengan menggunakan simulator ccnSim versi 2. ccnSim merupakan salah satu simulator jaringan khusus untuk jaringan CCN. Simulator ini mengimplementasi elemen-elemen utama dari CCN pada setiap *node* CCN, yaitu FIB, *Content Store* dan PIT. Pada ccnSim, *client* direpresentasikan sebagai sekumpulan *user* yang terhubung pada *node/router CCN*, yang melakukan *request* terhadap konten. *Request* pada konten mengikuti proses Poisson, dengan frekuensi yang dapat diatur.

Spesifikasi perangkat keras dan perangkat lunak yang digunakan pada penelitian ini adalah sebagai berikut:

- Hardware:
  - Processor: Intel Core i7 2,7 GHz
  - Memory: 4 GB 1600 MHz
- Software
  - Sistem Operasi: Ubuntu 12.04.05 lts - 64 bit
  - Simulator: ccnSim-02

Simulasi dilakukan pada 3 topologi sebagai lingkungan uji coba, yang merupakan topologi sintetis (topologi tiyok , topologi tree_6_2_modified) dan topologi riil (topologi abilene). Masing-masing topologi memiliki perbedaan jumlah *node*, jumlah *repository* yang aktif, konektivitas antar *node* dalam jaringan CCN, dan jarak antar *node* dengan *repository* yang beragam. Diagram topologi tercantum pada gambar 5.

Pada topologi pertama, topologi tiyok, seperti terlihat pada gambar 5b, merupakan topologi yang digunakan pada penelitian PCP [4]. Pada topologi ini terdapat dua *repository* sebagai *content store* jaringan CCN dan sembilan *node* yang terhubung dengan *client*. Dua *repository* pada topologi ini dibuat sedemikian rupa untuk menciptakan jarak/*distance* yang berbeda antara *node-node client* dengan *repository* yang berbeda.

Tabel I. Parameter simulator ccnSim yang digunakan untuk uji coba

Parameter	Keterangan	Nilai
network	topologi yang digunakan	{tiyok, tree_6_2, abilene}
node_repos	node yang menjadi repositori / *content store*	sesuai topologi
num_repos	jumlah node yang menjadi repositori / *content store*	sesuai topologi
node_clients	node yang terhubung dengan *client*	sesuai topologi
num_clients	jumlah node yang terhubung dengan *client*	sesuai topologi
lambda	jumlah *request* dari *node client* setiap detik	{1,10,20}
check_time	waktu simulator (satuan detik simulator) mengecek status download dari sebuah *node client*	0.1
file_size	ukuran file maksimum dari katalog (repositori) - dalam satuan *chunk*	{100,250,500}
objects	kardinalitas katalog	10^6
alpha	*MZipf shaping factor*	1.5
FS	*forwarding strategy*	spr
DS	*decision strategy*	prog (*progressive strategy*)
RS	algoritma *cache* yang digunakan	{LRUd, LRU}
C	ukuran *cache* dari *node*	10^4
ts	*time-sampling*, waktu pengambilan tiap sampel	0.1
partial_n	jumlah *node* yang diperlukan mencapai kestabilan *hit_rate* untuk memulai pengambilan statistik	-1 (semua *node* harus stabil, untuk memulai perhitungan statistik)
steady	durasi lamanya simulasi pengambilan statistik	1800 detik waktu simulator

Selanjutnya penggunaan topologi kedua (gambar 5a) sebagai lingkungan uji coba, yaitu topologi tree_6_2 yang dimodifikasi, digunakan sebagai pelengkap hasil uji coba dari topologi yang pertama, yaitu dengan jarak maksimal antara *node repository* dengan *node client* yang lebih besar, serta jumlah *node client* yang lebih banyak. Terdapat satu *node* yang bertindak sebagai *repository* / *content store*, dan 32 *node* yang terhubung dengan *client.*

Sedangkan topologi ketiga, topologi Abilene (gambar 5c), dipilih untuk mewakili bentuk topologi yang benar-benar nyata. Dalam penggunaan topologi ini terdapat 11 *node* CCN, yang konektivitas antar *node* sesuai dengan topologi Abilene, dan kecepatan link antar *node* berbeda-beda (sudah ditetapkan simulator). Topologi ini sudah tersedia dari simulator ccnSim.

Uji coba dilakukan selama 1800 detik waktu simulator, setelah jaringan CCN mencapai keadaan *cache* yang stabil. Pengecekan kestabilan *cache* dilakukan setiap 60 detik waktu simulator, dimana sampel *cache* tiap *node* diambil setiap 0.1 detik. Parameter simulasi secara detail dapat dilihat pada tabel I.

B. Hasil Uji Coba dan Analisa

Untuk mengukur kinerja metode yang kami usulkan, hasil uji coba metode *progressive cache* - LRUd dibandingkan dengan hasil uji coba metode *progressive cache* - LRU. Metrik pengukuran yang dibandingkan antara lain adalah sebagai berikut:

a) ***cache hit-rate***, rata-rata *cache hit-rate* setiap *node* yang ada pada topologi jaringan. *Cache hit-rate* setiap *node* didapat dari rumus sebagai berikut:

$$cache\ hit\text{-}rate = \frac{cache\ hit}{cache\ hit + cache\ miss} \quad (1)$$

dimana:
cache hit: jumlah *interest* yang dapat dipenuhi oleh sebuah *cache*
cache miss: jumlah *interest* yang tidak dapat dipenuhi dari sebuah *cache*

b) ***hdistance***, merupakan rata-rata *distance*/jarak dalam ukuran *hop*, yang harus ditempuh sebuah *interest* untuk mendapatkan data *chunk* dari *repository* terdekat. Rata-rata *hdistance* dapat dihitung berdasarkan rumus sebagai berikut:

$$hdistance = \frac{\sum d}{\sum interest} \quad (2)$$

dimana:

d: jarak tempuh data dari *repository* terdekat ke *requesting node (node* yang terhubung dengan *client* yang melakukan *request)*

interest: jumlah *interest* yang ada selama simulasi dijalankan (total *interest*)

c) ***download*** , merupakan total jumlah *download* yang berhasil diselesaikan selama simulasi berjalan

$$Total\ download = \sum_{i=0}^{n} download\ node_i \quad (3)$$

dimana:

download node$_{i:}$ jumlah download yang berhasil diselesaikan *node i*

Dari 3 metrik pengukuran yang dicatat, parameter *hit-rate* dan *hdistance* merupakan parameter yang digunakan untuk menilai kinerja sistem *cache* pada jaringan. Sedangkan parameter jumlah *download* yang berhasil dilakukan dalam periode tertentu merupakan metrik yang digunakan untuk menilai kinerja jaringan, yang dalam hal ini jaringan CCN.

Tabel II menunjukkan hasil perbandingan dari implementasi *progressive cache* dengan LRUd dan *progressive cache* dengan LRU. Dari hasil yang didapat dari simulasi, penggunaan *progressive cache* dengan LRUd dapat meningkatkan kinerja jaringan CCN pada ketiga metrik pengukuran. Peningkatan *hit-rate* maksimum sebesar 0.416% yang terjadi pada topologi tree_6_2_modified, dan efisiensi *hdistance* maksimum sebesar 1.145% yang terjadi pada topologi abilene. Sedangkan kinerja jaringan CCN keseluruhan, yang ditunjukkan oleh parameter total *download* yang berhasil dilakukan selama waktu simulasi, terjadi peningkatan kinerja dengan nilai maksimum 0.722%.

V. KESIMPULAN

Metode *replacement strategy* LRUd dikembangkan dengan tujuan meningkatkan kinerja strategi *replacement* metode *progressive cache* dalam jaringan *content centric networking* (CCN). Berdasarkan hasil uji coba yang didapat dari penelitian dengan menggunakan simulator ccnSim-2, penerapan *replacement strategy* LRUd pada metode *progressive cache* dalam jaringan CCN dapat meningkatkan kinerja jaringan. Penerapan metode ini dapat diterapkan pada segala jenis topologi jaringan CCN, tidak terbatas pada topologi jaringan yang tetap dan memiliki hirarki.

Ada beberapa hal yang bisa diteliti lebih lanjut, antara lain penambahan faktor frekuensi akses *chunk* untuk ikut diperhitungkan dalam *replacement strategy*, atau pemberian bobot untuk beberapa kandidat *chunk* (lebih dari dua) sebagai perhitungan dalam *replacement strategy*.

Tabel II. Hasil Uji Coba

Metrik Pengukuran	Topologi	Peningkatan Hasil (%)
peningkatan hit rate	tiyok	0.037
	tree_6_2_modified	0.416
	abilene	0.410
efisiensi hdistance	tiyok	0.598
	tree_6_2_modified	1.044
	abilene	1.145
peningkatan jumlah download	tiyok	0.722
	tree_6_2_modified	0.079
	abilene	0.264

DAFTAR PUSTAKA

[1] V.Jacobson, D. K. Smetters, J. D. Thornton M.F. Plass, N. H. Briggs, dan R.L. Braynard, 2009. "Networking Named Content", ser. CoNext '09, New york, NY, USA: ACM, 2009, pp.1-12.
[2] Cisco Visual Networking Index, 2010. Forecast and Methodology: 2010-2015
[3] L. Zhang, D.EStrin, J. Burk, V.Jacobson, J.D. Thornton, D. K. Smetters, B.Zhang, G. Tsudik, D. massey, C. Papadopulos, and et al., 2010. "Named Data Networking (ndn) project", October, pp. 1-26, 2010.
[4] J.M. Wang, B.Bensaou, 2012. "Progressive Caching in CCN*", Globecom 2012 – Next Generation Networking and internet symposium.
[5] H. Xu, Z. Chen, R. Chen, J. Cao, 2011. "Live streaming with content centric networking", 2012 Third International Conference on Networking and Distributed Computing, Hangzhou, China, 2012, pp. 1-5
[6] K. Katsaros, G. Xylomenos, and G. C. Polyzos, "MultiCache: An overlay architecture for information-centric networking," Computer Networks, pp. 1–11, 2011.
[7] S. Podlipnig and L Boszormenyi, 2003. "A Survey of Web *Cache* replacement Strategies", ACM Computing Surveys vol 35 no 4, December 2003.
[8] Krishnamurthy, B. and Rexford, J., 2001. "Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement", Addison-Wesley, Reading,MA , 2001.
[9] Hongfeng Xu, Zhen Chen, Rui Chen, Junwei Cao, 2011. "Life Streaming with Content Centric Networking".
[10] Rossi Dario, and Giuseppe Rossini., 2011. "Caching performance of content centric networks under multi-path routing (and more)." *Relatório técnico, Telecom ParisTech*.
[11] http://ots.utsystem.edu/pubs/internet2/4.html