Analisis Perbandingan Kinerja *Routing* Statis Pada *Named Data Networking* Berbasis *Software Defined Networking* Dan *Routing* Nlsr Pada Ndn Tradisional

1st Peiter Solarso Pasaribu
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
peiterspasaribu@student.telkomuniversity.ac.id

2nd Leanna Vidya Yovita
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
leanna@telkomuniversity.co.id

3rd Sofia Naning Hertiana
Fakultas Teknik Elektro
Telkom University
Bandung, Indonesia
sofiananing@telkomuniversity.ac.id

Abstrak—Named Data Networking (NDN) merupakan arsitektur jaringan baru yang menggunakan pendekatan content-centric, NDN diusulkan untuk menggantikan arsitektur saat ini yaitu arsitektur IP. Meskipun begitu, NDN masih memiliki beberapa permasalahan salah satunya NDN masih bergantung pada *Network Flooding* untuk menangani urusan routing dan forwarding. *Network Flooding* menjadi suatu permasalahan karena membanjiri informasi kesegala arah yang memberikan beban pada jaringan dan perangkat jaringan. Oleh karena itu, pada paper ini dilakukan integrasi NDN dengan SDN, SDN ini digunakan agar urusan *routing* dan *forwarding* tidak lagi bergantung pada *Network Flooding* dari NDN, melainkan menjadi urusan SDN dan untuk mengetahui perbandingan performa *routing* yang ditangani NDN dengan *routing* yang ditangani SDN. Evaluasi dilakukan dengan melakukan pengukuran performa *Retrieval Time*/*Round Trip Time* (RTT), *Throughput*, dan CPU *Usage* dengan *manual traffic* dan *generated traffic*. Dari hasil pengujian/pengukuran didapat, untuk aspek RTT arsitektur NDN memiliki waktu RTT lebih rendah ketika skenario topologi dibuat kompleks (tidak terdapat rute alternatif), namun ketika topologi linear NDN-SDN memiliki waktu lebih rendah. Sedangkan untuk aspek *throughput* dan CPU *usage* NDN-SDN lebih sering unggul daripada NDN itu sendiri.

Kata kunci— ***Named Data Networking (NDN), Software Defined Networking (SDN), manual traffic, generated traffic***

Abstract—Named Data Networking (NDN) is a new network architecture that uses a content-centric approach, NDN is proposed to replace the current architecture, namely IP architecture. Even so, NDN still has several problems, one of which is that it still relies on Network Flooding to handle routing and forwarding matters. Network Flooding becomes a problem because it floods information in all directions which puts a burden on the network and network devices. Therefore, in this paper, integration of NDN with SDN is carried out, SDN is used so that routing and forwarding matters no longer depend on Network Flooding from NDN, but become SDN matters and to compare the performance of routing handled by NDN with routing handled by SDN. Evaluation is done by measuring the performance of Retrieval Time/Round Trip Time (RTT), Throughput, and CPU Usage with manual traffic and generated traffic. From the test/measurement results obtained, for the RTT aspect of the NDN architecture, the RTT time is lower when the topology scenario is made complex (there are no alternative routes), but when the NDN-SDN linear topology has a lower time. Meanwhile, in terms of throughput and CPU usage, NDN-SDN is often superior to NDN itself.

Keywords— Named Data Networking (NDN), Software Defined Networking (SDN), manual traffic, generated traffic

I. PENDAHULUAN

Seiring berjalannya waktu, perkembangan internet terus menerus mengalami peningkatan dalam hal teknologi dan juga dalam penggunanya. Kebanyakan saat ini penggunaan internet ditujukan untuk mengakses sebuah konten dan konten jenis video lah yang nantinya dimasa depan akan mengambil persentase yang paling besar [1]. Karena permasalahan tersebut arsitektur Internet *Protocol* (IP) yang digunakan internet, secara bertahap mulai menunjukkan ketidak efektifan seperti peningkatan resource di jaringan IP yang mengakibatkan sulitnya untuk terus mendukung mobilitas [2]. Beberapa arsitektur sudah diusulkan salah satunya *Named Data Networking* (NDN) yang merupakan cabang dari arsitektur *Information Centric Networking* (ICN) yang menggantikan model IP dari *host-centric* menjadi *information-centric* [3].

NDN merupakan arsitektur jaringan yang menggunakan pendekatan *content- centric*, tidak

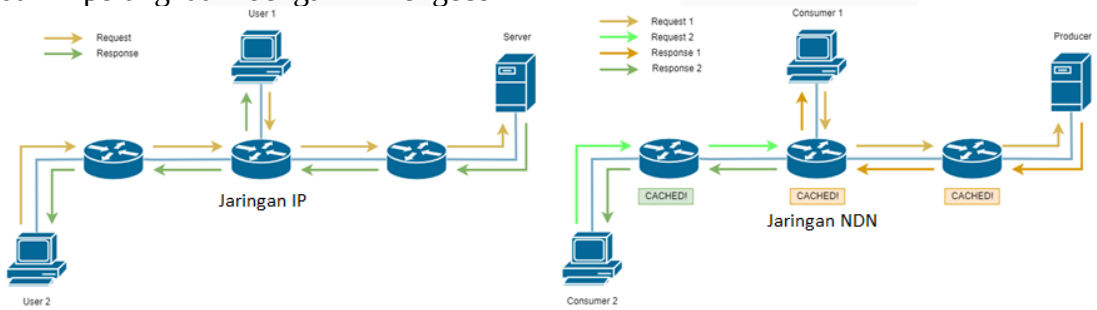
seperti arsitektur IP yang menggunakan pendekatan *host-centric*. Dengan pendekatan ini memberikan kemungkinan terpenuhinya tujuan utama dari NDN yaitu mengimplementasikan *network-caching* [4]. Namun *routing* pada arsitektur NDN masih bergantung pada *Network Flooding* ketika melakukan pengumuman perutean, dengan menyebarkan lokasi konten ke seluruh jaringan. Hal tersebut memberikan dampak beban pada sisi jaringan dan *resource* perangkat [1][5]. Dari permasalahan tersebut, diusulkanlah penggunaan Software Defined Networking (SDN) dalam penerapan NDN, dimana nantinya akan memisahkan control plane dan data plane, dengan begitu memungkinkan urusan *routing* dan *forwarding* ditangani oleh SDN controller [4].

Pada penelitian-penelitian sebelumnya [1][6][5], melakukan pengujian dari berbagai strategi *routing* (NLSR, NDN Flooding, dsb) yang hanya berfokus pada aspek *bootstrapping* time dan *cache hit ratio* dengan skenario topologi yang kurang bervariasi tidak memiliki *multiple route* / tanpa alternatif *route* dan hanya menggunakan *traffic* konstan bukan menggunakan *traffic* dari model distribusi. Oleh karena itu dalam tugas akhir ini akan dilakukan analisis perbandingan kinerja *routing* statis pada NDN berbasis SDN dan *routing* NLSR pada NDN tradisional, yang akan memfokuskan untuk mengetahui kinerja pada jaringan dan perangkat dengan mengecek parameter *Round Trip Time* (RTT) / *Retrieval Time* (RT), *Network Throughput*, CPU *Usage* dengan 15 menggunakan skenario yang lebih bervariasi (terdapat alternatif *route* dan *multi consumer* dengan Manual *Traffic* dan *Traffic* bermodel distribusi random zipf).

II. KAJIAN TEORI

A. Jaringan IP vs Jaringan NDN

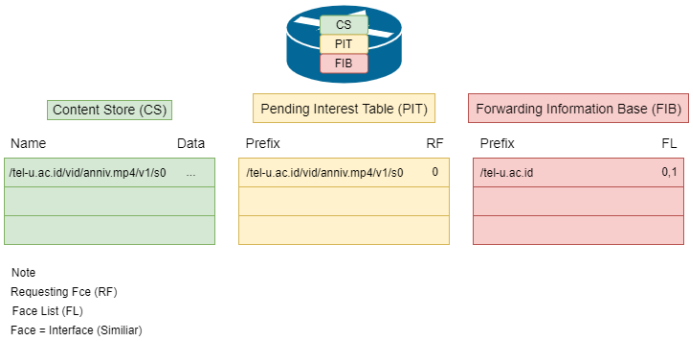
Pada dasarnya secara pendekatan yang ditawarkan jaringan IP dan jaringan NDN sendiri sudah menunjukkan perbedaan yang sangat kontras, dimana jaringan IP mengusulkan pendekatan host-centric sedangkan NDN mengusulkan pendekatan *content-centric* atau lebih tepatnya *data-centric*. Arsitektur Jaringan IP yang menggunakan pendekatan *host-centric* mengharuskan pertukaran data dilakukan berdasarkan alamat spesifik dalam hal ini berupa IP address. Sedangkan arsitektur NDN yang menggunakan pendekatan *data-centric*, proses pertukaran data akan didasarkan pada penamaan data hal tersebut membuat proses pertukaran data lebih fleksibel karena pengambilan data tidak diharuskan berdasarkan alamat spesifik namun bisa dari mana saja selagi data yang diminta sesuai [7]. Untuk lebih jelas dapat melihat pada **Gambar 2.1** memperlihatkan perbedaan proses pengambilan data pada Arsitektur IP dan NDN.



GAMBAR II.1
PERBANDINGAN ARSITEKTUR IP DAN ARSITEKTUR NDN

B. Named Data Networking (NDN)

Named Data Networking (NDN) merupakan sebuah arsitektur yang menggunakan pendekatan *content-centric* atau lebih tepatnya data-centric. Dengan pendekatan ini memberikan kemungkinan terpenuhinya tujuan utama dari NDN yaitu mengimplementasikan network-caching. Hal itu membuat pertukaran data tidak lagi berdasarkan alamat spesifik suatu *node*/*host* melainkan berdasarkan penamaan suatu data, pertukaran data dapat dilayani oleh *node*/*host* manapun selagi data yang diminta sesuai [4][8].

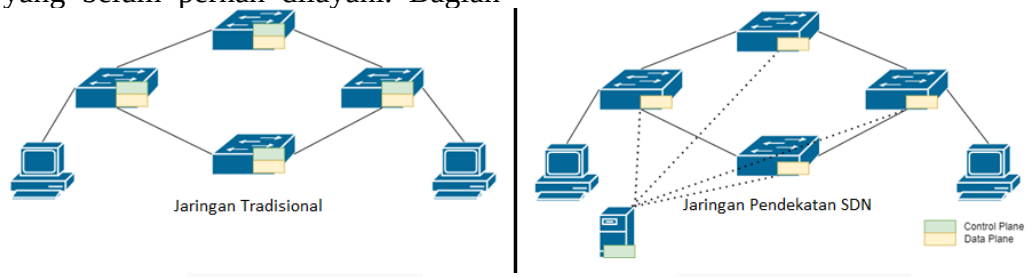


GAMBAR II.2
ARSITEKTUR NODE NDN

Pada **Gambar 2.2** menunjukkan arsitektur *node* NDN terdiri dari 3 bagian besar yaitu *Content Store* (CS), *Pending Interest Table* (PIT), dan *Forwarding Information Base* (FIB). Bagian CS berfungsi untuk menyimpan (meng-cache) baik itu sebagian maupun keseluruhan dari suatu data yang melewati node ini, tergantung dari strategi *caching* yang diterapkan. sedangkan bagian PIT berfungsi untuk mencatat permintaan-permintaan terhadap suatu data yang belum pernah dilayani. Bagian terakhir yaitu FIB berfungsi untuk menyimpan informasi jalur menuju *node* yang memiliki konten tertentu [8].

C. Software Defined Networking (SDN)

SDN merupakan sebuah pendekatan yang dipakai untuk mendesain, mengimplementasikan, dan memanajemen jaringan dengan cara memisahkan *network control* (*control plane*) dan *forwarding process* (data plane). Berkat pendekatan tersebut memunculkan berbagai manfaat khususnya pada fleksibilitas dan kemampuan kontrol jaringan. Dilain sisi memberikan manfaat kemudahan dalam memelihara dan memanage jaringan karena SDN memberikan kemampuan vendor-agnostic yang membebaskan kita menggunakan perangkat dari vendor apapun dengan begitu dapat mengurangi biaya, baik biaya perangkat maupun biaya teknisi [9]. Perbedaan antara jaringan tradisional dan jaringan SDN dapat dilihat pada **Gambar 2.3**.

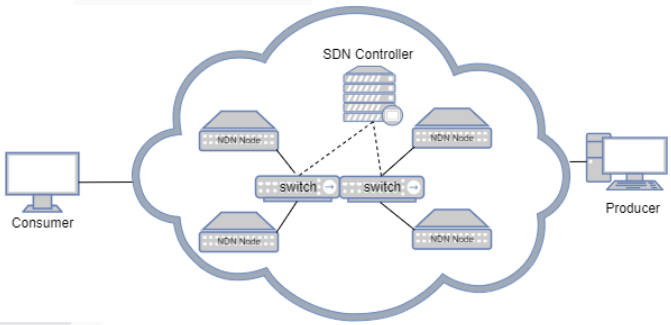


GAMBAR II.3
PERBEDAAN JARINGAN TRADISIONAL DAN SDN

III. METODE

A. Desain Sistem

Pada **Gambar 3.1** merupakan perancangan arsitektur Integrasi NDN dengan SDN akan dibentuk dari beberapa komponen diantaranya *consumer* (NDN node), *Intermediate node* (Switch dan NDN *node*), *producer* (NDN *node*), dan SDN *Controller*. *Consumer-Producer* akan bertindak sebagai peminta data dan pelayan data, sedangkan *Intermediate node* akan bertindak untuk meneruskan data berdasarkan rute yang dipilih, dan terakhir SDN *Controller* akan berfungsi untuk penguraian paket NDN agar dapat dibaca dan diteruskan pada arsitektur IP (Switch node).

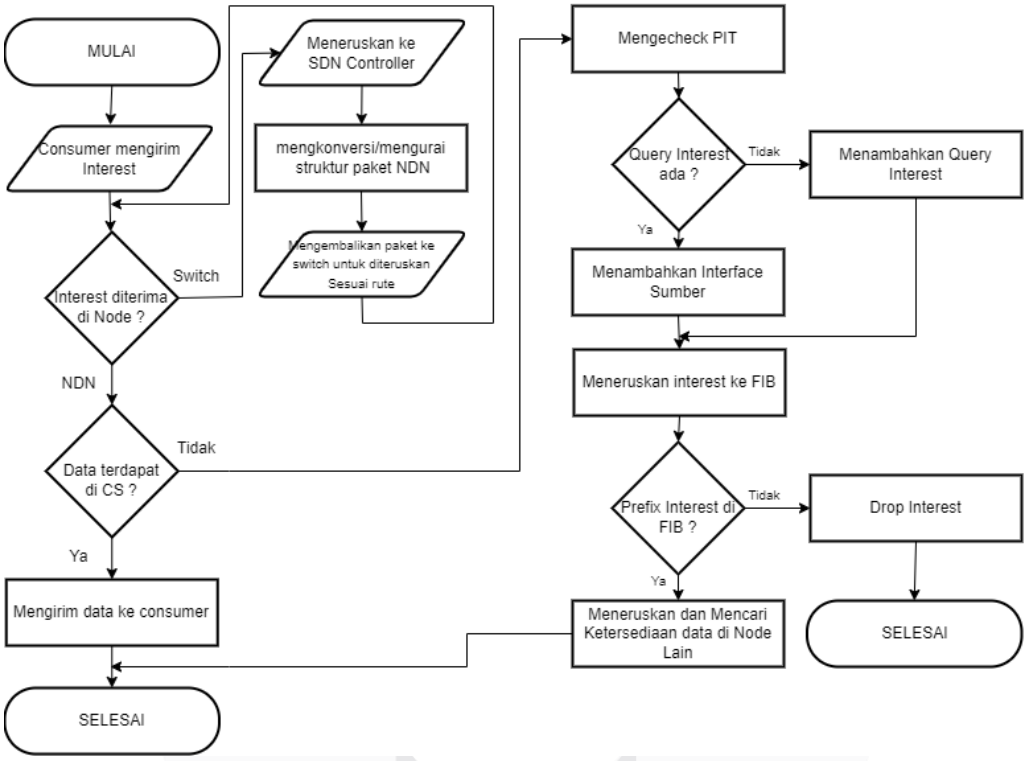


GAMBAR III.1
DESAIN SISTEM

B. Diagram Alir Sistem

Pada **Gambar 3.2** menjelaskan alur kerja dari sistem Arsitektur NDN yang diintegrasikan dengan SDN. Ketika sebuah *consumer* mengirimkan permintaan terhadap sebuah paket, maka akan mengirimkan sebuah *interest* ke *node* di depannya. Jika *interest* tersebut diterima oleh *switch*, maka *interest* tersebut akan diteruskan ke SDN *Controller* untuk diuraikan agar dapat

dibaca data dan tujuannya kemudian *interest* tersebut akan dikembalikan ke *switch* tadi untuk selanjutnya diteruskan ke node berikutnya sesuai dengan rute yang telah ditentukan SDN *Controller*. Disisi lain, jika *interest* diterima oleh *node* NDN maka pertama tama akan mengecek ketersediaan data di *Content Store* (CS) apabila data ditemukan maka data akan dikirimkan ke *node* pengirim *interest*. jika data tidak ditemukan di CS, maka interest akan diteruskan ke PIT untuk mengecek ketersediaan *query interest* tersebut. Apabila *query* tidak ditemukan maka akan menambahkan *query* terkait *interest* tersebut, namun jika ditemukan maka hanya akan menambahkan *interface* sumber *interest* ke *query* tersebut. Selanjutnya setelah melalui PIT, *interest* akan diteruskan ke FIB untuk mencari rute terhadap *interest* tersebut. Jika terdapat *prefix interest* di FIB maka *interest* akan diteruskan ke rute tersebut untuk melakukan pencarian data dengan proses yang sama secara berulang seperti sebelumnya. Namun jika *prefix interest* tidak ditemukan di FIB maka akan di *drop* dan permintaan terhadap data tidak berhasil dilayani.



GAMBAR III.2
DIAGRAM ALIR KERJA

C. Skenario Pengujian

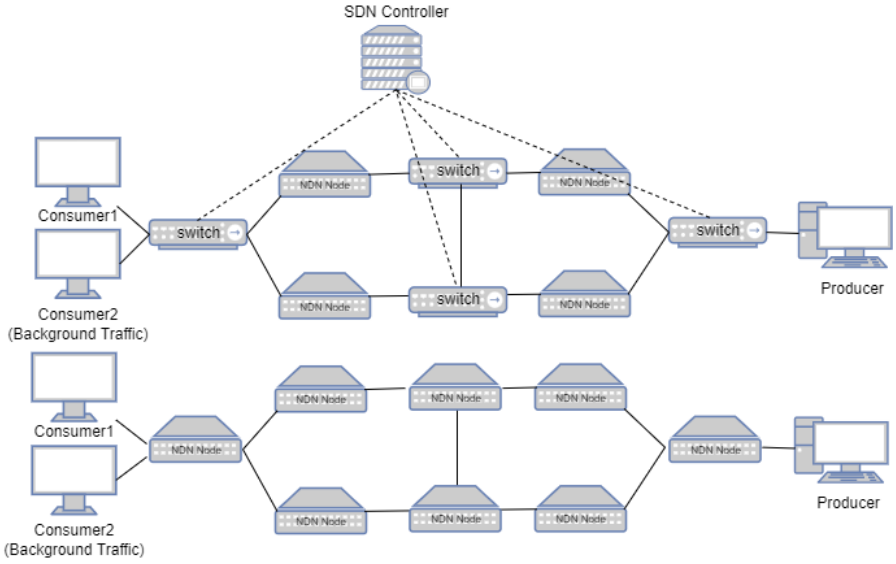
Pada bagian ini, akan dilakukan pengujian untuk mengukur kinerja dari strategi *routing* statis pada NDN berbasis SDN dengan *routing* NLSR pada NDN tradisional, dengan aspek pengujian sebagai berikut :

1. *Round Trip Time (RTT) / Retrieval Time* : Waktu yang dibutuhkan dari pengiriman permintaan terhadap data hingga data diterima.
2. *Network Throughput* : Mengacu pada tingkat pengiriman pesan yang berhasil melalui saluran komunikasi.
3. CPU *Usage* : Persentase penggunaan CPU.
4. *Manual Traffic* dan *Generated Traffic* : Percobaan terhadap parameter pengujian dilakukan dengan dua cara yaitu dengan mengirimkan paket/data secara manual dan dengan bantuan *Generated Traffic* untuk membuat dan mengirimkan *Traffic* secara lebih bervariasi.

1. Skenario Topologi Bercabang

Pada skenario ini akan dilakukan pengujian kinerja dari strategi routing statis (arsitektur NDN-SDN) dan NLSR (arsitektur NDN tradisional) terhadap kinerja jaringan dan perangkat jaringan. Pengujian kinerja dilakukan dengan cara melihat aspek RTT, *throughput*, dan terakhir CPU *usage* dari *node* yang dilewati paket.

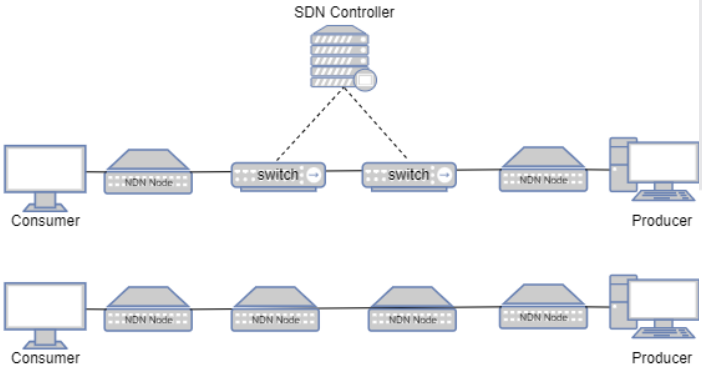
Untuk topologi yang digunakan, akan terlihat seperti pada **gambar 3.3** dimana terdapat 7 buah *node* NDN (4 *intermediate node*, 1 *producer node*, 2 *consumer node*), 4 buah *switch*, 1 buah SDN *controller* (Ryu). *Consumer* pertama melakukan request paket dan dimonitoring sembari *consumer* kedua melakukan *request* paket sebagai *background traffic*.



GAMBAR III.3
SKENARIO TOPOLOGI BERCABANG

2. Skenario Topologi Linear

Pada skenario ini akan dilakukan pengujian kinerja dari strategi routing statis (arsitektur NDN-SDN) dan NLSR (arsitektur NDN tradisional) terhadap kinerja jaringan dan perangkat jaringan. Pengujian kinerja dilakukan dengan cara melihat aspek RTT, *throughput*, dan terakhir CPU *usage* dari *node* yang dilewati paket. Untuk topologi yang digunakan, akan terlihat seperti pada **gambar 3.4** dimana terdapat 4 buah *node* NDN (2 *intermediate node*, 1 *producer node*, 1 *consumer node*), 2 buah *switch*, 1 buah SDN *controller* (Ryu).



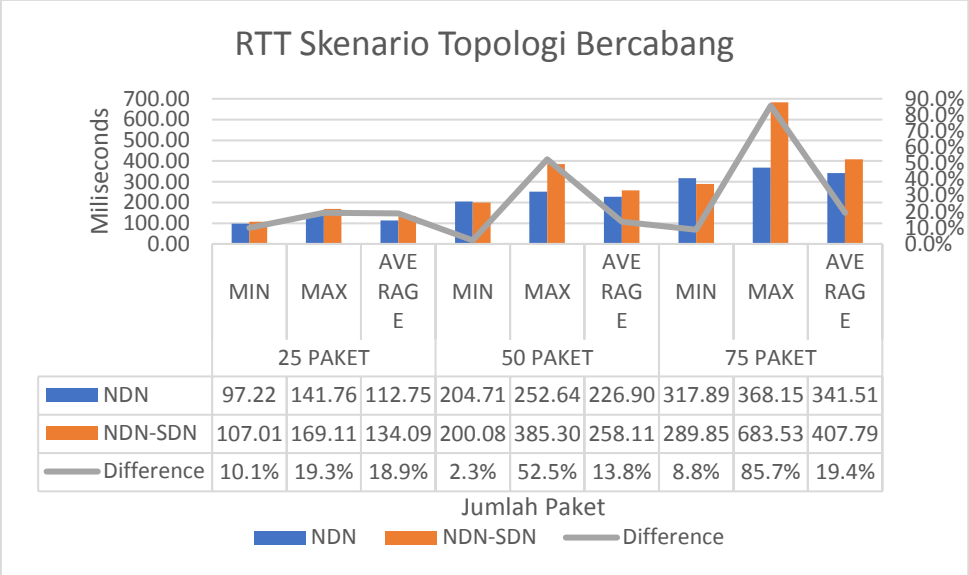
GAMBAR III.4
SKENARIO TOPOLOGI LINEAR

IV. HASIL DAN PEMBAHASAN

A. Pengukuran Round Trip Time (RTT)

1. Skenario Topologi Bercabang

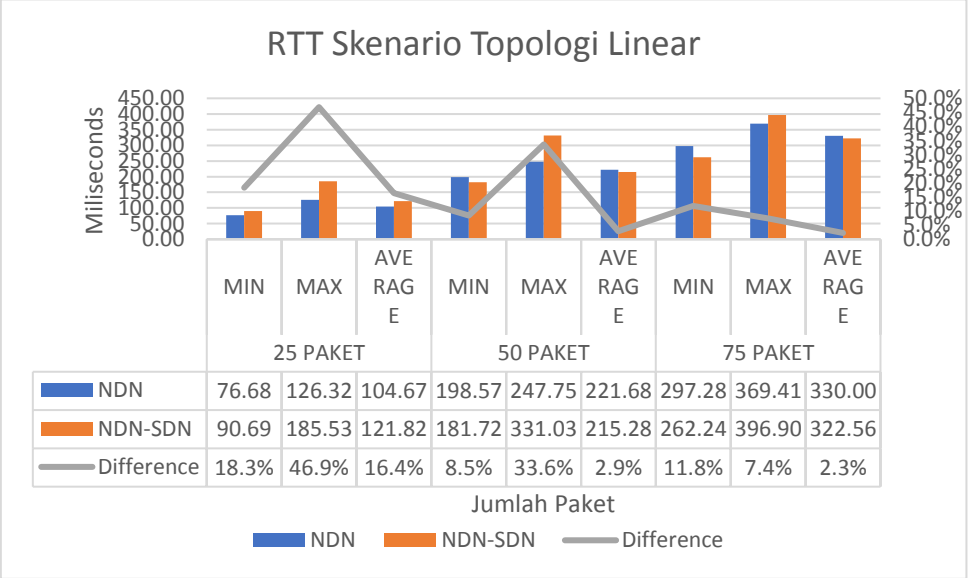
Pada **Gambar 4.1** menunjukan hasil pengukuran RTT pada Skenario Topologi Bercabang. Dari grafik menunjukkan hasil arsitektur NDN yang menggunakan routing NLSR memperoleh waktu RTT lebih rendah dengan kisaran perbedaan antara 13.8% hingga 19.4% di keseluruhan 3 group paket (Pada perolehan waktu rata-rata), jika dibandingkan dengan arsitektur NDN-SDN yang menggunakan routing statis.



GAMBAR IV.1
HASIL RTT SKENARIO TOPOLOGI BERCABANG

2. Skenario Topologi Linear

Pada **Gambar 4.2** menunjukan hasil pengukuran RTT pada Skenario Topologi Linear. Dari grafik menunjukkan hasil arsitektur NDN-SDN yang menggunakan routing statis memperoleh waktu RTT lebih tinggi pada pengiriman 25 paket, Namun ketika jumlah paket dibesarkan menjadi 50 paket dan 75 paket, NDN-SDN memperoleh waktu RTT rata-rata (*average*) lebih rendah jika dibandingkan dengan arsitektur NDN yang menggunakan routing NLSR.



GAMBAR IV.2
HASIL RTT SKENARIO TOPOLOGI LINEAR

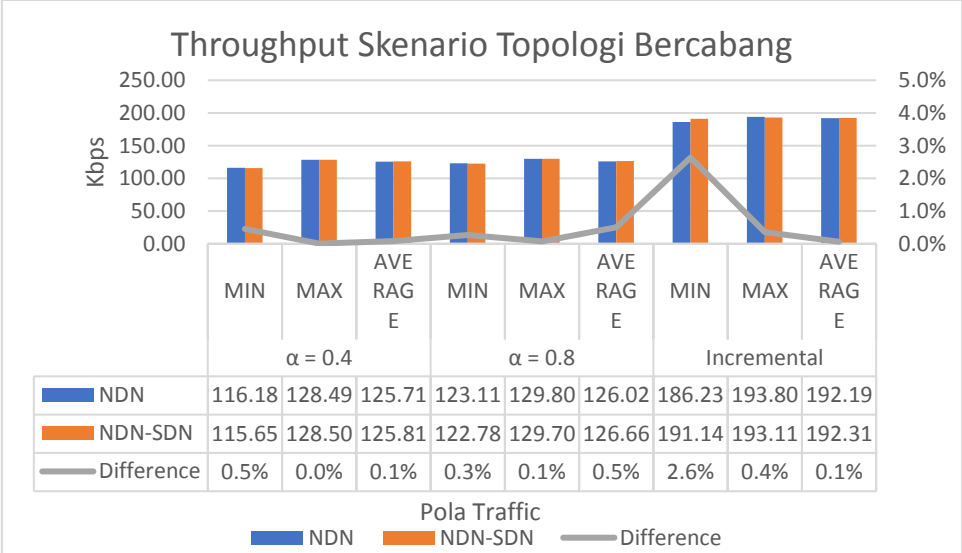
3. Analisis Hasil Pengukuran RTT

Berdasarkan data hasil pengujian RTT untuk keseluruhan skenario topologi (Skenario topologi bercabang dan topologi linear), menunjukkan hasil yang bagus untuk arsitektur NDN tradisional dikarenakan memiliki waktu RTT yang lebih sedikit dibandingkan dengan NDN-SDN (Pada skenario topologi bercabang), Namun hasil yang berbeda didapat pada skenario topologi linear yang menunjukkan arsitektur NDN-SDN memiliki waktu RTT lebih sedikit dibandingkan NDN tradisional. Perbedaan hasil antara kedua skenario topologi ini dikarenakan skenario topologi bercabang disusun dengan cara meletakkan switch secara terpisah serta memiliki link alternatif, tidak seperti skenario topologi linear yang peletakan

switchnya saling bersebelahan dan tidak ada link alternatif sama sekali. Alhasil pada skenario topologi bercabang arsitektur NDN-SDN mendapatkan waktu RTT yang lebih lama dikarenakan perlu melakukan penguraian struktur paket NDN secara berulang dan terjadi proses pembandingan *forwarding* tabel dari setiap switch di sisi SDN *Controller*. Namun untuk skenario topologi linear arsitektur NDN-SDN mendapatkan waktu RTT lebih kecil dikarenakan proses penguraian paket NDN hanya dilakukan sekali dan proses pembandingan forwarding tabel berjalan lebih cepat karena query pada *forwarding* tabel lebih sedikit (hal ini dikarenakan tiap switch hanya memiliki 2 *node* tetangga tidak seperti di skenario topologi bercabang).

B. Pengukuran Throughput

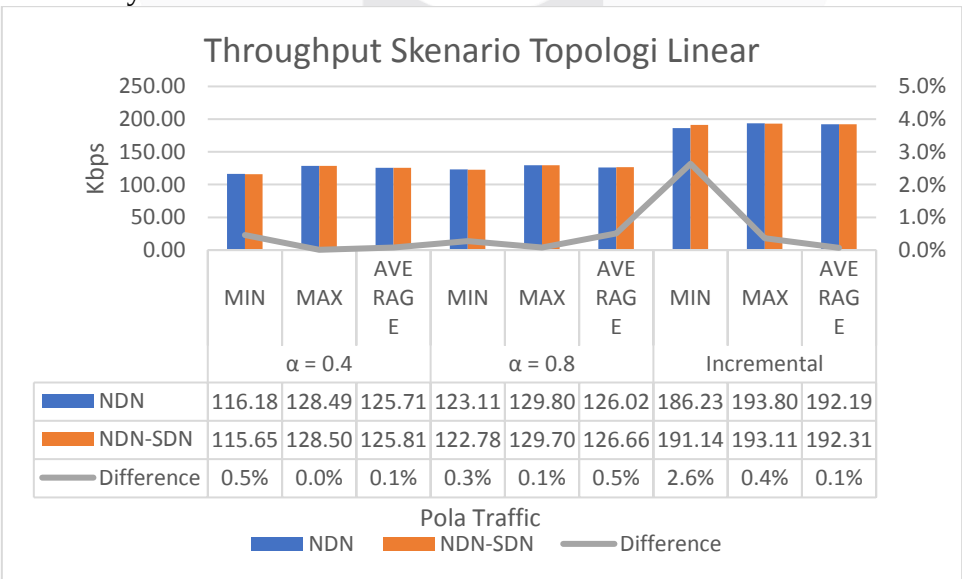
1. Skenario Topologi Bercabang



GAMBAR IV.3
HASIL THROUGHPUT SKENARIO TOPOLOGI BERCABANG

Pada **Gambar 4.3** menunjukan hasil pengukuran RTT pada Skenario Topologi Bercabang. Dari grafik menunjukkan hasil arsitektur NDN-SDN yang menggunakan routing statis memperoleh waktu RTT lebih tinggi meskipun perbedaan hanya dikisaran 0.X% di keseluruhan 3 group paket (Pada perolehan *throughput* rata-rata), jika dibandingkan dengan arsitektur NDN yang menggunakan routing NLSR.

2. Skenario Topologi Linear



GAMBAR IV.4
HASIL THROUGHPUT SKENARIO TOPOLOGI LINEAR

Pada **Gambar 4.4** menunjukan hasil pengukuran RTT pada Skenario Topologi Bercabang. Dari grafik menunjukkan hasil arsitektur NDN-SDN yang menggunakan routing statis memperoleh waktu RTT lebih tinggi meskipun perbedaan hanya dikisaran 0.X% di keseluruhan 3 group paket (Pada perolehan *throughput* rata-rata), jika dibandingkan dengan arsitektur NDN yang menggunakan routing NLSR.

3. Analisis Hasil Pengukuran Throughput

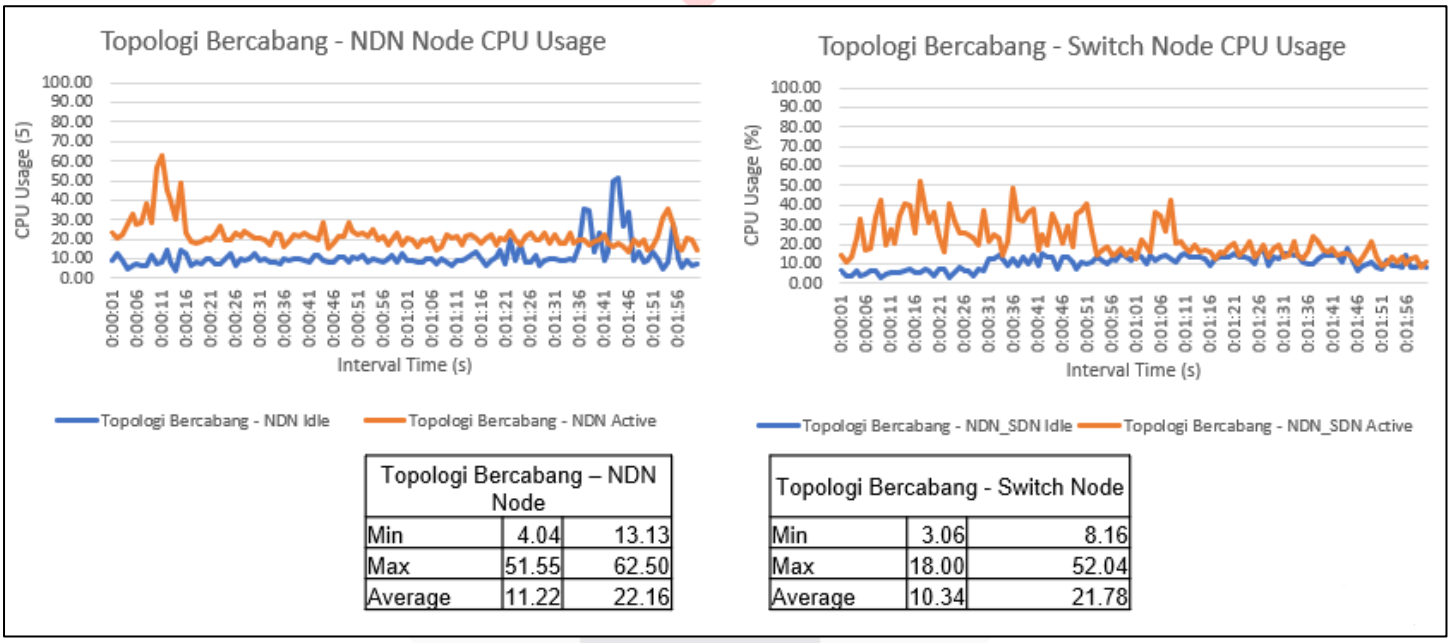
Berdasarkan data hasil pengujian *throughput* untuk keseluruhan skenario topologi (skenario topologi bercabang dan linear), di ketiga pola trafik pada masing-masing skenario topologi menunjukkan penggunaan *Switch node* lebih berpeluang menghasilkan throughput yang lebih tinggi dibandingkan dengan NDN *node* itu sendiri (meskipun perbedaannya sangat kecil hanya kisaran 0.X%), hal ini dikarenakan kapasitas *bandwidth* pada *interface*/*link switch node* selalu lebih tinggi daripada kapasitas NDN node, seperti yang tampak pada **Gambar 4.5**.

```
mini-ndn> iperf s1 s2
*** Iperf: testing TCP bandwidth between s1 and s2
*** Results: ['9.55 Gbits/sec', '9.56 Gbits/sec']
mini-ndn> iperf s1 s2
*** Iperf: testing TCP bandwidth between s1 and s2
*** Results: ['13.1 Gbits/sec', '13.1 Gbits/sec']
mini-ndn> iperf s1 s2
*** Iperf: testing TCP bandwidth between s1 and s2
*** Results: ['11.6 Gbits/sec', '11.6 Gbits/sec']
mini-ndn> iperf n5 pr
*** Iperf: testing TCP bandwidth between n5 and pr
*** Results: ['2.21 Gbits/sec', '2.21 Gbits/sec']
mini-ndn> iperf n5 pr
*** Iperf: testing TCP bandwidth between n5 and pr
.*** Results: ['2.34 Gbits/sec', '2.34 Gbits/sec']
mini-ndn> iperf n5 pr
*** Iperf: testing TCP bandwidth between n5 and pr
*** Results: ['1.40 Gbits/sec', '1.40 Gbits/sec']
mini-ndn>
```

GAMBAR IV.5
KAPASITAS BANDWIDTH LINK NDN (N5-PR) DAN SWITCH (S1-S2)

C. Pengukuran CPU Usage

1. Skenario Topologi Bercabang



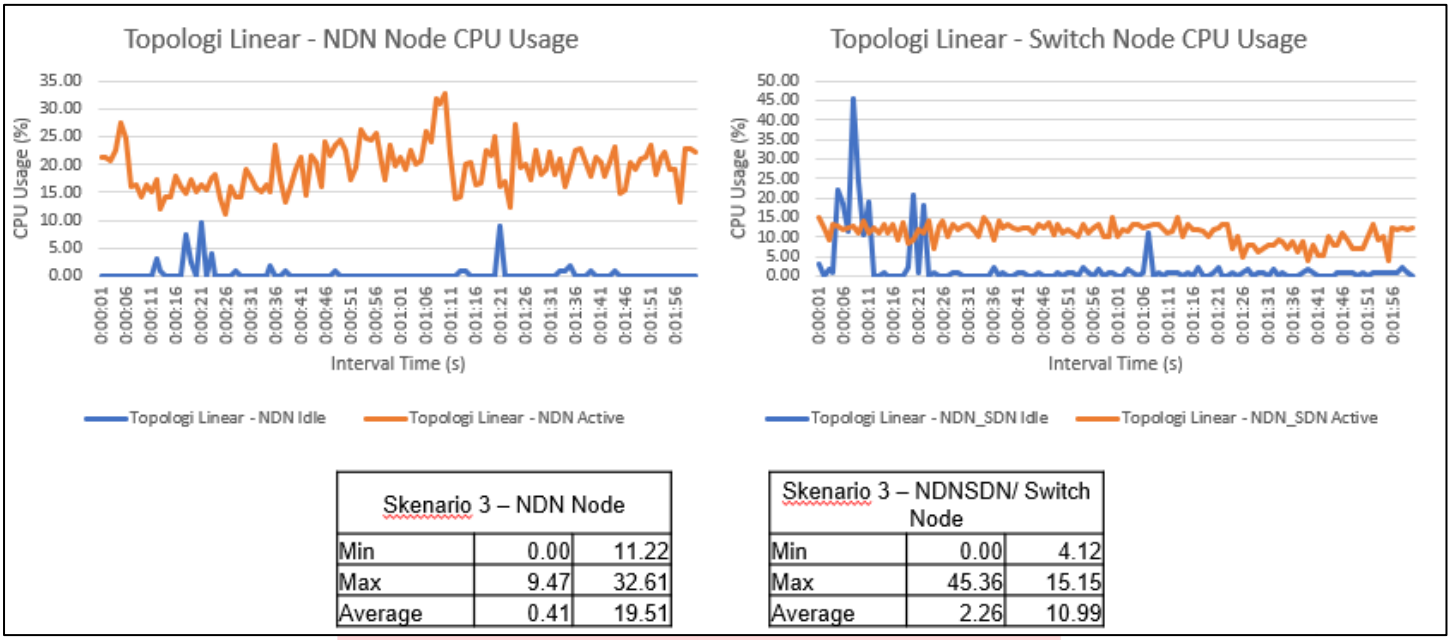
Topologi Bercabang – NDN Node		
Min	4.04	13.13
Max	51.55	62.50
Average	11.22	22.16

Topologi Bercabang - Switch Node		
Min	3.06	8.16
Max	18.00	52.04
Average	10.34	21.78

GAMBAR IV.6
KONSUMSI CPU NDN NODE (ARSITEKTUR NDN) DAN CPU SWITCH NODE (ARSITEKTUR NDN-SDN)

Pada **Gambar 4.6** menunjukan hasil pengukuran CPU *Usage* pada Skenario Topologi Bercabang. Dari grafik menunjukkan hasil konsumsi CPU pada *switch node* (arsitektur NDN-SDN) yang menggunakan routing statis memperoleh konsumsi CPU lebih rendah baik pada saat consumer aktif maupun consumer tidak aktif dibandingkan dengan konsumsi CPU NDN node (arsitektur NDN) yang menggunakan routing NLSR.

2. Skenario Topologi Linear



Skenario 3 – NDN Node		
Min	0.00	11.22
Max	9.47	32.61
Average	0.41	19.51

Skenario 3 – NDNSDN/ Switch Node		
Min	0.00	4.12
Max	45.36	15.15
Average	2.26	10.99

GAMBAR IV.7
KONSUMSI CPU NDN NODE (ARSITEKTUR NDN) DAN CPU SWITCH NODE (ARSITEKTUR NDN-SDN)

Pada **Gambar 4.7** menunjukan hasil pengukuran CPU *Usage* pada Skenario Topologi Linear. Dari grafik menunjukkan hasil konsumsi CPU pada *switch node* (arsitektur NDN-SDN) yang menggunakan routing statis memperoleh konsumsi CPU Agak lebih tinggi pada saat status consumer tidak aktif yaitu 0.41% (rata-rata per detik konsumsi CPU NDN node) berbanding 2.26% (rata-rata per detik konsumsi CPU Switch node), namun ketika consumer dalam keadaan aktif melakukan request data, penggunaan CPU NDN node menjadi sangat lebih tinggi dibandingkan dengan Switch node yaitu rata-rata 19.51% per detik dibandingkan dengan rata-rata 10.99% per detik.

3. Analisis Hasil Pengukuran CPU Usage

Berdasarkan data hasil pengujian CPU *Usage* untuk keseluruhan skenario topologi (Skenario Topologi Bercabang dan Skenario Topologi Linear), keseluruhan skenario menunjukkan hasil yang bagus untuk arsitektur NDN-SDN karena *switch node* memiliki konsumsi CPU yang sangat lebih kecil dibandingkan dengan NDN *node*. *Switch node* memiliki konsumsi CPU yang sangat kecil dikarenakan proses penguraian paket, informasi ketersediaan rute/node tetangga, dan proses kalkulasi pemilihan rute diteruskan ke SDN *controller* untuk ditanganinya, jadi switch hanya difungsikan untuk meneruskan paket berdasarkan hasil kalkulasi SDN *Controller*. Dilain sisi ketika hanya arsitektur NDN semata, semua permasalahan tersebut (proses penguraian, kalkulasi pemilihan rute, dll) ditangani di tiap node NDN itu sendiri, hal tersebutlah yang menyebabkan konsumsi CPU NDN *node* lebih besar dibandingkan dengan *switch node*.

V. KESIMPULAN

Dari hasil pengujian dan pengukuran memperlihatkan bahwa arsitektur NDN-SDN yang menggunakan routing statis memiliki kemampuan pengiriman yang lebih cepat jika topologi bersifat sederhana hanya berupa topologi linear (tanpa adanya rute bercabang) dan dengan syarat lain switch node harus disusun berdampingan, namun hasil yang berbeda akan didapat bila topologi bersifat kompleks (terdapat rute bercabang) dan *switch node* disusun terpisah. Perbedaan hasil tersebut dikarenakan semakin banyak rute alternatif semakin banyak pula informasi *forwarding table switch* yang perlu diproses dan dibandingkan satu sama lain dan berdampak pada semakin lamanya proses pengiriman. Namun ketika topologi bersifat linear NDN-SDN lebih cepat melakukan pengiriman dibandingkan NDN karena proses penguraian/pembacaan paket di SDN Controller berjalan lebih cepat daripada proses penguraian/pembacaan paket di NDN *node*. Meskipun *bandwidth link switch node* lebih tinggi yang memungkinkan throughput menjadi lebih tinggi, tetapi lamanya proses pengiriman paket

juga dipengaruhi oleh beberapa *delay* jaringan salah satunya yaitu *processing delay* (proses penguraian paket di NDN *node* maupun SDN *Controller*). Dilain sisi untuk aspek CPU *usage*, *switch node* memiliki konsumsi yang lebih rendah dibandingkan NDN *node*, karena semua tugas kalkulasi ditangani oleh SDN *Controller*.

REFERENSI

[1] A. Kalghoum and S. M. Gammar, "Towards new Information Centric Networking strategy based on software defined networking," *IEEE Wirel. Commun. Netw. Conf. WCNC*, 2017, doi: 10.1109/WCNC.2017.7925536.

[2] X. Guo, N. Liu, X. Hou, S. Gao, and H. Zhou, "An Efficient NDN Routing Mechanism Design in P4 Environment," *2021 2nd Inf. Commun. Technol. Conf. ICTC 2021*, pp. 28–33, 2021, doi: 10.1109/ICTC51749.2021.9441639.

[3] M. Amadeo, C. Campolo, G. Ruggeri, A. Molinaro, and A. Iera, "Understanding Name-based Forwarding Rules in Software-Defined Named Data Networking," *IEEE Int. Conf. Commun.*, vol. 2020-June, 2020, doi: 10.1109/ICC40277.2020.9149266.

[4] M. Alhowaidi, D. Nadig, B. Ramamurthy, B. Bockelman, and D. Swanson, "Multipath Forwarding Strategies and SDN Control for Named Data Networking," *Int. Symp. Adv. Networks Telecommun. Syst. ANTS*, vol. 2018-Decem, pp. 1–6, 2018, doi: 10.1109/ANTS.2018.8710068.

[5] J. V. Torres, O. Carlos, and M. B. Duarte, "CRoS-NDN: Controller-based Routing Strategy for Named Data Networking".

[6] E. Aubry, T. Silverston, and I. Chrisment, "SRSC: SDN-based routing scheme for CCN," *1st IEEE Conf. Netw. Softwarization Software-Defined Infrastructures Networks, Clouds, IoT Serv. NETSOFT 2015*, 2015, doi: 10.1109/NETSOFT.2015.7116130.

[7] Y. N. Rohmah, D. W. Sudiharto, and A. Herutomo, "The performance comparison of forwarding mechanism between IPv4 and Named Data Networking (NDN). Case study: A node compromised by the prefix hijack," *Proceeding - 2017 3rd Int. Conf. Sci. Inf. Technol. Theory Appl. IT Educ. Ind. Soc. Big Data Era, ICSITech 2017*, vol. 2018-Janua, pp. 302–306, 2017, doi: 10.1109/ICSITech.2017.8257129.

[8] L. Zhang *et al.*, "Named data networking," *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014, doi: 10.1145/2656877.2656887.

[9] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," *Secur. Commun. Networks*, vol. 9, no. 18, pp. 5803–5833, 2016, doi: 10.1002/sec.1737.