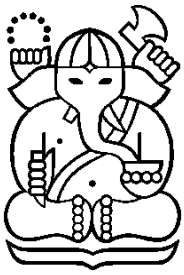
PROTOKOL DAN ARSITEKTUR PERUTEAN UNTUK MENDUKUNG SKALABILITAS DAN MOBILITAS PADA *NAMED DATA NETWORK*

DISERTASI

Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Doktor dari
Institut Teknologi Bandung

Oleh
TODY ARIEFIANTO WIBOWO
NIM: 33217012
(Program Studi Doktor Teknik Elektro dan Informatika)



INSTITUT TEKNOLOGI BANDUNG
Agustus 2023

ABSTRAK

PROTOKOL DAN ARSITEKTUR PERUTEAN UNTUK MENDUKUNG SKALABILITAS DAN MOBILITAS PADA *NAMED DATA NETWORK*

Oleh
Tody Ariefianto Wibowo
NIM: 33217012
(Program Studi Doktor Teknik Elektro dan Informatika)

Perkembangan teknologi internet saat ini, dikhawatirkan akan membawa jaringan kepada fasa kolaps, disebabkan pergeseran sifat dari internet sendiri saat ditemukan hingga karakteristik penggunaannya kini. *Named Data Network* (NDN) membawa paradigma baru dan memberikan perubahan mendasar dalam implementasi komunikasi data di internet. Perutean adalah merupakan salah satu faktor penting dalam mekanisme pengiriman pesan pada NDN dan skalabilitas pada perutean, dalam rangka melayani jumlah user dan konten yang semakin besar, merupakan masalah yang timbul dengan semakin banyaknya entitas di internet.

Penelitian NDN dimulai dari Simulasi yang merupakan metode yang menuntut sumber daya terkecil, kemudian terus membesar kearah emulasi dan testbed. Testbed dengan perangkat prototipe paling mendekati realitas dunia nyata namun makin sulit dilakukan. Diperlukan sebuah kerangka kerja yang dapat mendekatkan penelitian NDN ke dunia nyata dengan upaya dan sumber daya yang lebih terkontrol

Pada tahap pertama, protokol perutean yang diusulkan diberi nama *Grid-based Scalable NDN Routing* (GSNR). Dimana sebelumnya dibangun terlebih dahulu model matematis efisiensi protokol perutean geografis dan link state pada NDN. Pada penelitian tahap kedua diusukan kerangka kerja yang menggabungkan keunggulan emulasi dan testbed. Kerangka kerja tersebut melibatkan proses emulasi NDN-PnetLab (PL) dengan menggunakan VM berbasis Qemu untuk pendekatan implementasi yang lebih mudah, pemantauan yang komperhensif, dan efektivitas dari sisi biaya.

GSNR, dengan menggunakan protokol perutean hibrid, memberikan performa yang lebih baik dan stabil dibandingkan protokol perutean sebelumnya NLSR dan GEO. GSNR mampu memberikan *message overhead* yang rendah tanpa mengorbankan performa transimis data. GSNR menunjukan performa yang stabil untuk *throughput* dan *delay*, dengan percobaan perubahan luas area dan jumlah *node*. Performa GSNR (*overhead, path-stretch, delay, throughput*), secara rata-rata, hanya mengalami perubahan (meningkat atau berkurang) 19.83% dibandingkan 116.46% pada NLSR dan 50.91% pada GEO.

Hasil pengamatan menunjukkan bagaimana kerangka kerja NDN-PnetLab (PL) yang diusulkan dapat memantau parameter QoS, penggunaan CPU, dan penggunaan RAM. Penggunaan CPU dan RAM tidak mungkin untuk diperiksa dan dianalisis jika kita hanya menggunakan NDNSim atau bahkan Mini-NDN. Dibandingkan dengan Mini-NDN, kerangka kerja PL memberikan perbedaan performa 7,59% pada RTT delay, dan 0,89% pada nCHR, kemudian dapat memberikan hasil pengukuran parameter yang intensif terkait performa hardware. Hal ini sangat dibutuhkan untuk melanjutkan penelitian ke implementasi sistem NDN.

Kata kunci: named data network, perutean, klusterisasi, mobilitas, emulasi.

ABSTRACT

ROUTING PROTOCOL AND ARCHITECTURE FOR SKALABILITY AND MOBILITY SUPPORT IN NAMED DATA NETWORK

By
Tody Ariefianto Wibowo
NIM: 33217012
(Doctoral Program in Electrical Engineering and Informatics)

Development of internet technology today, has been woried will bring the network to collapse, due to shifting nature of the internet itself since created from nowadays characteristic. Named Data Network (NDN) brings a new paradigm and provides fundamental changes in the implementation of data communications on the internet. Routing is one of the important factors in messaging delivery mechanism on NDN and the scalability in perutean is an issue that arises with the increasing number of entities on the internet.

NDN research starts from Simulation which is a method that demands the smallest resources, then continues to expand towards emulation and testbeds. Testbeds with prototype devices are closest to real world reality but are increasingly difficult to implement. A framework is needed that can bring NDN research closer to the real world with more controlled efforts and resources.

In the first part, the proposed routing protocol is named Grid-based Scalable NDN Routing (GSNR). Where previously a mathematical model for the efficiency of geographic routing protocols and link state was built on the NDN. In the second part of the research, a framework is proposed that combines the advantages of emulation and testbed. The framework involves the NDN-PnetLab (PL) emulation process using Qemu-based VMs for an easier implementation approach, comprehensive monitoring, and cost-effectiveness.

GSNR, by using a hybrid routing protocol, provides better and more stable performance than the previous routing protocols NLSR and GEO. GSNR is able to provide low message overhead without sacrificing data transmission performance. GSNR shows stable performance for throughput and delay, with experiments scenarios, changing the area and the number of nodes. GSNR performance (overhead, path-stretch, delay, throughput), on average, only changed (increased or decreased) 19.83% compared to 116.46% in NLSR and 50.91% in GEO.

The results of the observations show how the proposed NDN-PnetLab (PL) framework can monitor QoS parameters, CPU usage, and RAM usage. CPU and RAM usage is impossible to check and analyze if we only use NDNSim or even Mini-NDN. Compared to Mini-NDN, the PL framework provides a performance

difference of 7.59% on RTT delay, and 0.89% on nCHR, then can provide intensive parameter measurement results related to hardware performance. This is urgently needed to continue research into the implementation of NDN systems.

Keywords: named data network, routing, clustering, mobility, emulation.

PROTOKOL DAN ARSITEKTUR PERUTEAN UNTUK MENDUKUNG SKALABILITAS DAN MOBILITAS PADA *NAMED DATA NETWORK*

Oleh
Tody Ariefianto Wibowo
NIM: 33217012
(Program Studi Doktor Teknik Elektro dan Informatika)

Institut Teknologi Bandung

Menyetujui
Tim Pembimbing

Tanggal 7 Agustus 2023

Ketua	Anggota
(Prof. Dr. Nana Rachmana Syambas)	**(Ir. Hendrawan, M.Sc., Ph.D)**

PEDOMAN PENGGUNAAN DISERTASI

Disertasi Doktor yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada penulis dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin penulis dan harus disertai dengan kaidah ilmiah untuk menyebutkan sumbernya.

Sitasi hasil penelitian Disertasi ini dapat di tulis dalam bahasa Indonesia sebagai berikut:

Tody, A.W, Syambas N.R, dan Hendrawan. (2023): Protokol dan Arsitektur Perutean Untuk Mendukung Skalabilitas dan Mobilitas Pada *Named Data Network,* Disertasi Program Doktor, Institut Teknologi Bandung.

dan dalam bahasa Inggris sebagai berikut:

Tody, A.W, Syambas N.R, and Hendrawan. (2023): *Routing Protocol and Architecture for Skalability and Mobility Support in Named Data Network*, Doctoral Dissertation, Institut Teknologi Bandung.

Memperbanyak atau menerbitkan sebagian atau seluruh disertasi haruslah seizin Dekan Sekolah Pascasarjana, Institut Teknologi Bandung.

Dipersembahkan kepada Orang-tua, Istri, Putra-putriku, Kakak- adikku, Mertua serta Keluarga besarku tercinta yang senantiasa mendukung langkah dan perjuanganku.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT, karena atas karuniaNya penulis dapat menyelesaikan buku Disertasi ini. Selanjutnya salawat dan salam penulis haturkan untuk junjungan kita Muhammad SAW. Banyak sekali pengalaman dan pembelajaran berharga selama penulis menempuh studi S3 ini.

Ucapan terima kasih penulis ucapkan untuk istri dan anak-anak yang telah mendukung penulis sepenuhnya dalam menyelesaikan studi doktoral ini sehingga dapat dilaksanakan dengan baik.

Terima kasih tak lupa penulis sampaikan untuk Prof. Dr. Nana Rachmana Syambas selaku promotor, yang telah membimbing, memberikan segala masukan berharga dan dukungan yang sangat besar sehingga penulis dapat melaksanakan studi ini dengan baik dan dapat memperoleh pengalaman riset yang berharga di Jepang. Terima kasih untuk Ir. Hendrawan, M.Sc., Ph.D yang telah membimbing, mengoreksi dan memberikan masukan yang sangat berharga bagi penulis selama melaksanakan studi ini. Terima kasih untuk seluruh tim penguji untuk semua tahap dalam pengerjaan disertasi ini, Prof. Dr. Carmadi Machbub, Dr. Ian Joseph Matheus Edrward, Dr. Iskandar, Dr. Tutun Juhana, dan Prof. Dr. Rendy Munadi untuk segala masukan berharga bagi penelitian penulis.

Semoga karya ini dapat memberikan manfaat bagi semua pihak.

Bandung, Agustus 2023

DAFTAR ISI

DAFTAR GAMBAR DAN ILUSTRASI

DAFTAR TABEL

DAFTAR SINGKATAN DAN LAMBANG

SINGKATAN	Nama	Pemakaian pertama kali pada halaman
AS	*Autonomous System*	2
ASICs	*Application-specific Integrated Circuit chips*	3
BGP	*Border Gateway Protocol*	3
CCND	*Content-Centric Networking Daemon*	13
CDN	*Content Delivery Network*	1
CS	*Content Store*	7
FIB	*Forward Information Base*	7
GPSR	*Greedy Perimeter Stateless Routing*	19
GRE	*Generic Routing Encapsulation*	14
HR	*Routing Hiperbolik*	18
IAB	*Internet Architecture Board*	2
ICN	*Information-Centric Networking*	1
IETF	*Internet Engineering Task Force*	1
IP	*Internet Protocol*	1
ISOC	*Internet Society*	2
IS-IS	*Intermediate System*	2
NDN	*Named Data Network*	1
NDN-IoT	*Named Data Network – Internet of Things*	4
NLSR	*Named Data Link State Routing Protocol*	3
NRS	*Name Routing System*	20
OLSA	*Opaque Link-State Advertisements*	14
OSPF	*Open Shortest Path First*	2
OSPFD	*OSPF Daemon*	13
OSPFN	*OSPF for Named-data*	12
PIT	*Pending Interest Table*	7
PKI	*Public Key Infrastructure*	20
RIP	*Routing Information Protocol*	2
SDN	*Software Defined Network*	20
WSN	*Wireless Sensor Network*	17

Bab I
Pendahuluan

I.1 Latar Belakang

Pencarian konten/data pada jaringan berbasis IP sesungguhnya berfokus pada pembentukan koneksi bukan mencari konten/data. Hal tersebut dikatakan oleh Van Jacobson sebuah "*misconduct of internet's behavior*". Mekanisme IP tersebut diprediksi akan menyebabkan kolaps pada jaringan internet pada masa depan dalam waktu dekat, disebabkan pembangkitan trafik yang sangat besar pada proses komunikasi diatas. Sehingga dengan tujuan untuk memperbaiki mekanisme IP dalam mencari konten/data maka *Named Data Networking* (NDN) pertama kali diusulkan oleh Van Jacobson (Jacobson et al., 2009), (Zhang et al., 2014).

Dalam menangani permasalahan tersebut, beberapa teknik telah diusulakan, dengan mengedepankan ide penggunaan jaringan secara efisien, salah satu ide yang telah diimplementasikan secara luas adalah *Content Delivery Network* (CDN) (Krishnamurthy and Wang, 2000; *RFC 3040 - Internet Web Replication and Caching Taxonomy*, 2001). Paradigma dari teknologi ini adalah bagaimana konten dapat dibawa sedekat mungkin kepada pelanggan/klien, dengan cara membuat replika *server* penyedia konten. Dengan menggunakan CDN maka pencarian terhadap konten cukup disediakan oleh *server-server* CDN yang terdekat dari klien, sehingga hal ini dapat meningkatkan efisiensi pada utilitas jaringan. Namun demikian Van Jacobson melihat solusi CDN masih belum cukup, dibutuhkan perubahan paradigma yang lebih besar dikarenakan pada teknologi CDN pelanggan masih berfokus pada koneksi end to end walaupun konten telah berpindah lebih dekat ke pelanggan. Van Jacobson mengusulkan NDN sebagai solusi jaringan pada masa depan.

Tabel I.1 Tabel Perbandingan IP dan NDN

	IP	NDN
Penamaan	Nomor IP	URI (Uniform Resource Identifier)

Setup komunikasi	*Client – Server* atau *end to end*	*Client* – Penyedia konten atau berorientasi konten
Perutean & *Forwarding*	Klien ke *Server*	Klien ke *In network caching*, Produser pilihan terakhir
Node perantara	Meneruskan paket berdasarkan alamat IP	Mencari keberadaan konten terdekat berdasarkan nama

Perbedaan IP dan NDN sangatlah signifikan karena terjadi perubahan fundamental dalam paradigma komunikasi data yang selama ini terjadi sebagaimana dapat dilihat pada Tabel I.1. Orientasi IP terhadap komunikasi merupakan komunikasi klien ke *server* sedangkan NDN orientasinya adalah mencari konten terdekat dimana juga dilengkapi dengan mekanisme *caching* pada node-node perantara (*in network caching*). Node perantara pada jaringan IP hanya meneruskan paket berdasarkan alamat IP sedangkan node perantara pada NDN mencari keberadaan konten terdekat berdasarkan nama konten dengan format URI.

NDN sebagai bagian *Information-Centric Networking* (ICN) inisiatif, kelompok kerja bentukan IETF, membawa perubahan besar terhadap mekanisme internet, tidak saja membawa konten lebih dekat kepada pelanggan (konten berjarak satu hop) namun juga mengusulkan perubahan mekanisme dari berbasis alamat IP menjadi berbasiskan nama konten. Pada NDN jaringan akan memiliki mekanisme untuk mengutamakan membagikan konten jika konten sudah pernah diminta oleh pelanggan lain. *Server*/produser merupakan pilihan terakhir dalam menyediakan konten pada saat jaringan tidak memiliki konten yang diminta oleh pelanggan.

Disisi lain perubahan IP menjadi nama konten atau yang biasa disebut nama prefix menyebabkan peningkatan kompleksitas pada manajemen rute pada tiap *node* di NDN. Jika sebelumnya tiap *node* pada jaringan IP hanya melakukan pengaturan rute berdasarkan kelompok jaringan (biasa disebut network atau sub-netrwork), dengan nama prefix maka *node* harus melakukan pengaturan untuk masing-masing konten yang ada di jaringan (Yi et al., 2013). Sehingga masalah perutean dan

penerusan paket menjadi cukup kompleks pada mekanisme NDN sehingga dipandang perlu untuk mengkaji lebih lanjut terkait mekanisme tersebut.

Yi, et al. pada (Yi et al., 2013) telah menunjukkan bahwa mekanisme *forwarding* saja, tanpa mekanisme perutean, pada *router* NDN dapat menangani kegagalan link secara efektif. Strategi *forwarding* pada jaringan NDN memiliki fitur untuk menangani kegagalan link yang sebelumnya ditangani oleh mekanisme perutean pada jaringan IP. Hal tersebut menimbulkan pertanyaan, "Apakah router NDN masih membutuhkan mekanisme perutean?".

Yi, et al. menjalankan penelitian lanjutan pada (Yi et al., 2014) telah menjawab pertanyaan terkait kebutuhan perutean pada NDN. Pada penelitian lanjutan ini didapatkan bahwa mekanisme *forwarding* dalam penelitian sebelumnya telah gagal untuk (i) menentukan peringkat pilihan jalur pada fase awal jaringan (ii) memulihkan kembali pilihan jalur yang sudah dihilangkan sebelumnya. Penelitian ini juga mengusulkan untuk meningkatkan stabilitas dan skalabilitas mekanisme perutean dengan mengatasi kegagalan link yang berumur pendek, yang sering terjadi pada jaringan yang dinamis akibat pergerakan node. Hal tersebut menjadi justifikasi bahwa perutean pada NDN masih menjadi area penelitian yang menarik.

Perbedaan signifikan antara perutean pada NDN dan perutean pada IP adalah terkait subjek yang menjadi fokus pada protokol peruteannya. Pada IP, perutean memperhitungkan cost antara node-node pada alamat tertentu. Sedangkan pada NDN, perutean memperhitungkan cost pada node-node yang menyimpan konten diinginkan. Pada IP, node-node (server dan router) tidak memperhatikan konten karena IP hanya membaca alamat IP sumber dan tujuan. Pengiriman informasi hanya memperhatikan node dengan alamat IP yang dituju. Sedangkan pada NDN, setiap pengiriman informasi memperhatikan konten. Mencari konten dengan keterjangkauan paling baik merupakan tujuan dari perutean NDN. Sehingga algoritma peruteannya harus lebih smart dan memperhitungkan konten tidak hanya yang ada di produser/server namun juga konten yang terdapat pada node-node di sepanjang jaringan yang disimpan pada *content store*-nya.

I.2 Masalah Penelitian

Saat ini perutean pada IP merupakan salah satu teknik yang bisa dikatakan telah mapan, hal ini dibuktikan dengan dapat terhubungnya semua entitas internet di dunia. Dengan mudah kita dapat terhubung dari entitas satu ke entitas lain, walaupun secara geografis bisa saja keduanya terpisah amat jauh, selama keduanya merupakan bagian dari jaringan internet. Namun kondisi "mapan" di jabarkan oleh *Internet Architecture Board* (IAB), sebuah komite sebagai gabungan dari *Internet Engineering Task Force* (IETF) dan *Internet Society* (ISOC), dalam laporannya pada tahun 2007 (Meyer, 2007), bahwa perutean pada internet bermasalah pada skalabilitasnya. Lalu bagaimana dengan NDN?

Usulan terhadap protokol perutean NDN pada penelitian sebelumnya belum ada yang berfokus pada masalah skalabilitas, terutama terkait perutean *overhead* dan ketika berhadapan dengan lingkungan dengan jumlah user masif dan dinamis seperti pada IoT. Skalabilitas pada masalah perutean adalah terkait bagaimana mekanisme perutean dapat tetap berjalan baik seiring bertambahnya ukuran/skala jaringan, atau secara sederhana, perutean dapat menangani puluhan, ratusan, ribuan, atau ratusan juta *node* pada jaringan NDN. Tentu saja permasalahan akan semakin kompleks berkali lipat untuk jaringan yang juga memiliki sifat dinamis. Berdasarkan mekanisme yang ada di NDN paling tidak ada 3 hal yang akan berpengaruh pada skalabilitas perutean NDN yaitu: (i) efisiensi penamaan, (ii) jumlah *record* perutean tabel (iii) paket *overhead*. Efisiensi penamaan akan berhubungan erat dengan regulasi dan kemampuan komputasi pada *node* NDN. Jumlah record perutean tabel akan berhubungan dengan efisiensi harga dari sebuah *node* NDN, karena memory untuk perutean tabel biasanya menggunakan skema perangkat keras *Application-specific Integrated Circuit chips* (ASICs). Sedangkan paket *overhead*, yang akan menjadi salah satu fokus penelitian, merupakan tarik menarik antara kompleksitas dan performa.

Penelitian pada NDN juga telah bergerak kearah *node* NDN dengan kemampuan mobilitas. Tentu hal ini berhubungan erat dengan karakteristik dari jaringan tanpa kabel, *Internet of Things* (IoT), *ad-hoc*, dll yang memiliki karakter tersebut. NDN

sebagai teknologi pengganti IP tentu harus dapat menangani tantangan yang ada pada skema jaringan tersebut. *Node* yang bergerak menyebabkan mekanisme perutean harus dapat beradaptasi terhadap perubahan tersebut yang tentu saja berimplikasi terhadap pilihan jalur terbaik dalam pengiriman data. Pada dasarnya mekanisme perutean pada IP seperti berbasis *link state* atau *distance vector* tidak diciptakan untuk beradaptasi pada lingkungan mobile. Sehingga perutean pada NDN yang menggunakan mekanisme yang sama seperti NLSR (Wang et al., 2018) akan memiliki karakter yang sama. Lehman, dkk (Lehman et al., 2016) telah memperlihatkan jumlah *node* akan meningkatkan *overhead* perutean secara linier hingga 335 link, dan selebihnya, peningkatan *overhead* mendekati eksponensial.

Selain masalah perutean selanjutnya seperti lazimnya penelitian dan teknologi baru yang sedang berkembang, penelitian NDN juga memiliki cukup banyak halangan dalam penerapannya. Teknologi NDN jika ditarik dari teknologi jaringan IP lebih bersifat revolusioner dibanding penguatan. Hal ini dapat dilihat bahwa pada saat jaringan NDN dapat diimplementasikan secara mandiri maka kita tidak lagi memerlukan teknologi jaringan sebelumnya terutama pada spesifik lapisan IP. Penelitian NDN pada sisi simulasi dan emulasi memiliki gap yang cukup besar untuk berjalan ke arah implementasi. Sehingga untuk menjembatani hal tersebut dibutuhkan metode yang dapat mempermudah para peneliti untuk lebih dekat kearah implementasi NDN.

Dalam mendukung disertasi ini, juga telah dilakukan beberapa penelitian dan publikasi untuk memberikan fundamental penelitian. Survei terkait protokol perutean pada jaringan NDN telah dilakukan dan dipublikasikan pada (Ariefianto and Syambas, 2017). Telah dijabarkan apa saja kekurangan dan kelebihan dari berbagai jenis protokol perutean yang telah diusulkan pada jaringan NDN. Pada (Wibowo et al., 2018, 2019, 2020) telah menjabarkan, memodelkan dan memetakan peluang penelitian pada masalah skalabilitas pada protokol perutean. Salah satu usulan yang muncul dari pemetaan pada publikasi di atas adalah usulan menggunakan protokol perutean hibrid dan mekanisme grid pada arsitektur perutean.

Terkait dengan penjabaran tersebut maka dapat dituliskan bahwa diperlukan solusi untuk masalah penelitian sebagai berikut:

1. Belum ada algoritma perutean pada NDN yang menitik beratkan pada skalabilitas jaringann terutama *overhead* perutean dan *delay*.
2. *Overhead* perutean dan *delay* akan semakin meningkat seiring jumlah *node* yang masif dan mobilitas yang tinggi pada lingkungan IoT.
3. Jeda penelitian NDN antara penelitian hulu ke arah hilirisasi atau implementasi.

I.3 Tujuan dan Ruang Lingkup Penelitian

Tujuan umum dari penelitian ini adalah menghasilkan skema atau algoritma perutean pada jaringan NDN-IoT yang memiliki skema perutean yang efisien (perutean *overhead* rendah, perutean konvergensi yang baik, *delay* yang rendah, dan paket *delevery ratio* yang tinggi).

Tujuan khusus dari penelitian ini adalah:

1. Mengusulkan algoritma perutean yang lebih baik dalam hal mendukung skalabilitas dalam mengakomodir ukuran lingkungan *network* yang semakin besar, jumlah node yang semakin banyak dan pengaruh node yang bergerak.
2. Mengintegrasikan mekanisme grid pada algoritma perutean tersebut untuk menekan angka *overhead* dan meningkatkan performa.
3. Memodelkan efisiensi pada algoritma perutean
4. Mengusulkan metode penelitian NDN kearah implementasi

Ruang lingkup permasalahan dalam penelitian ini adalah:

1. Jaringan NDN yang diteliti adalah jaringan NDN-IoT, dimana setiap *node* memiliki kemampuan mobilitas.
2. Skema pengiriman data dapat melibatkan multi-hop, sehingga *node* yang terlibat dalam pengiriman adalah, (i) *node* produser sebagai *node* yang memiliki/memproduksi konten; (ii) *node* konsumer sebagai *node* yang

meminta konten, dan (iii) *node-node* lainnya yang berperan sebagai *node* tengah, antara produser dan konsumer.

3. Model popularitas permintaan mengikuti distribusi *Zipf Mandelbrot*.
4. Indikator kinerja yang dianalisis adalah *overhead* perutean, *delay*, effisensi, *troughput* dan *path stretch*.

I.4 Metodologi Penelitian

Metodologi penelitian dilakukan dengan tahapan sebagai berikut:

1. Studi literatur terkait NDN dan mengkaji peluang penelitian terkait skalabilitas perutean. Memetakan protokol perutean eksisting serta menelaah kelebihan dan kekurangan protokol perutean yang ada. Mencari peluang dari protokol perutean untuk dapat menangani kedinamisan jaringan dan dapat menagani jumlah node yang banyak pada area yang luas
2. Pengujian awal terkait perutean pada NDN.
3. Membuat model jaringan NDN lebih khusus terkait proses perutean sehingga dapat dibandingkan. Membangun model efisiensi algoritma perutean sehingga diketahui efisiensi dari algoritma perutean *link state* dan geografis
4. Membuat usulan formulasi protokol perutean hibrid dengan kluster dengan tujuan meningkatkan skalabilitas perutean namun memiliki mekanisme yang sederhana dan *low overhead*. Untuk menyederhanakan proses protokol perutean yang diusung menggunakan beberapa konsensus diawal untuk mengurangi beban koordinasi.
5. Menguji performansi algoritma perutean dan grid yang diusulkan pada poin 4 untuk melihat optimalisasi pada skalabilitas
6. Menguji coba metode pendekatan implementasi NDN yang dapat digunakan untuk implementasi algoritma usulan
7. Mengambil kesimpulan yang dilakukan dengan melihat hasil simulasi dan analisis algoritma perutean serta pendekatan implementasi testbed.

I.5 Asumsi Penelitian

Asumsi yang digunakan pada penelitian ini adalah sebagai berikut:

1. Setiap *node* NDN yang terlibat memiliki pengetahuan terhadap lokasi dan posisi dirinya.
2. Mekanisme perutean dipelihara secara terdistribusi
3. Proses transmisi atau pengiriman data di layer fisik selalu berjalan dengan baik. Permasalahan hanya dapat terjadi di sisi mekanisme perutean saja.
4. Proses *forwarding* yang ada mendukung aturan perutean yang diajukan.
5. Proses *caching* yang ada mendukung aturan perutean yang diajukan.

I.6 Hipotesis

Hipotesis pada disertasi ini adalah bahwa dengan menggunakan usulan protokol perutean hibrid berbasis Geografis dan Link State, mekanisme protokol perutean akan makin efisien. Pada perutean berbasis geografis, broadcast pesan perutean dilakukan dengan mekanisme reaktif dan pada area yang lebih kecil/kluster sehingga dapat memberikan message *overhead* yang minimal. Hal tersebut dapat tetap dipertahankan baik dengan adanya penambahan kepadatan node maupun juga cakupan area jaringan yang lebih luas. Sedangkan untuk perutean intra kluster protokol perutean bekerja dengan algoritma proaktif link state sehingga dapat memberikan performa komunikasi yang baik pada area yang luas.

Algoritma perutean hibrid yang diusulkan akan membatasi *broadcast* pesan perutean sehingga memberikan *message overhead* yang effisen tanpa mengorbankan performansi terkait dengan parameter *delay*, *throughput* dan *path stretch* untuk penambahan kepadatan *node* dan cakupan area jaringan yang lebih luas. Lebih jauh algoritma ini juga dapat menjaga performansi sistem walaupun *node-node*nya bergerak atau berpindah tempat.

Pendekatan metode implementasi dengan menggunakan *virtual machine* akan dapat meniru perilaku *node* NDN seperti menggunakan perangkat keras yang terdedikasi. Namun metode virtualisasi akan menghemat biaya dan tetap memudahkan dalam hal manajemen *node*.

I.7 Kebaruan dan Orisinalitas

Kebaruan dan orisinalitas pada penelitian disertasi ini adalah protokol perutean yang efektif dalam hal melakukan *broadcast overhead message*. Protokol perutean ini mengombinasikan kelebihan algoritma *routing* geografis dengan algoritma *rouitng* berbasiskan *link state*.

Penelitian dalam disertasi ini dibagi menjadi 3 bagian, dan secara lebih rinci dapat dijelaskan sebagai berikut:

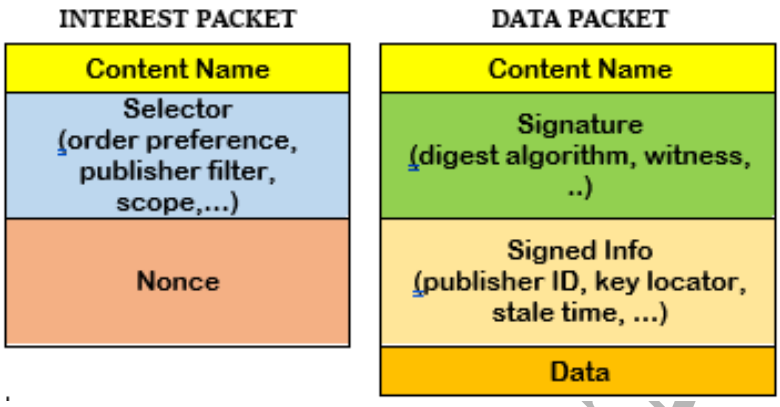
1. Pada bagian 1 dibuat model efisiensi
2. Pada bagian 2 dibuat protokol peruten hibrid
3. Pada bagian 3 dibuat model pendekatan implementasi NDN

Bab II

State of the Art Perutean pada *Named Data Network*

II.1 Arsitektur *Named Data Network*

Pada jaringan NDN, terdapat dua jenis paket yang beredar di jaringan, yaitu Interest Packet dan Data Packet (Jacobson et al., 2009). Paket *Interest* berisi informasi terkait konten yang diminta oleh konsumer. Data *Packet* adalah paket yang berisi data konten sesuai informasi *interest* yang dikirimkan dari produser (asal mula sebuah konten) ataupun *node* (biasa juga disebut *router*) lainnya yang diminta oleh konsumer (entitas yang meminta sebuah konten). Data *Packet* merupakan respon dari paket *Interest* yang dikirimkan sebelumnya. Struktur paket tersebut dapat dilihat pada Gambar II.1.

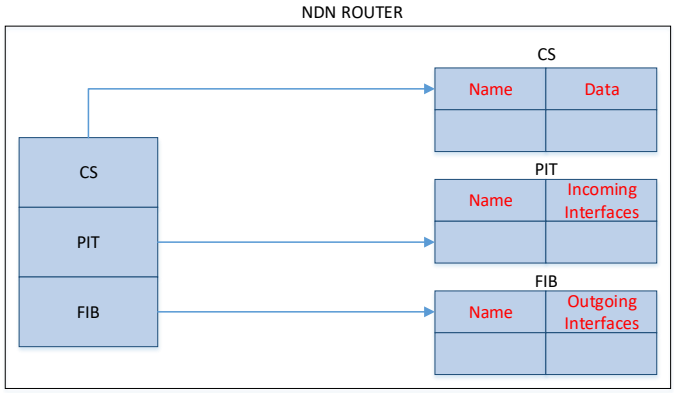


Gambar II.1 Tipe paket pada NDN (Jacobson et al., 2009; Zhang et al., 2014)

Bagian *Content Name* pada Paket *Interest* dan Paket Dat serta *Nonce* merupakan elemen yang harus ada, sedangkan bagian Selector merupakan bagian opsional yang digunakan untuk menyeleksi konten apa yang diinginkan oleh konsumer. Nonce berisi 4 *octet byte string* yang digenerate secara random ("Interest Packet — NDN Packet Format Specification 0.2-2 documentation," n.d.).

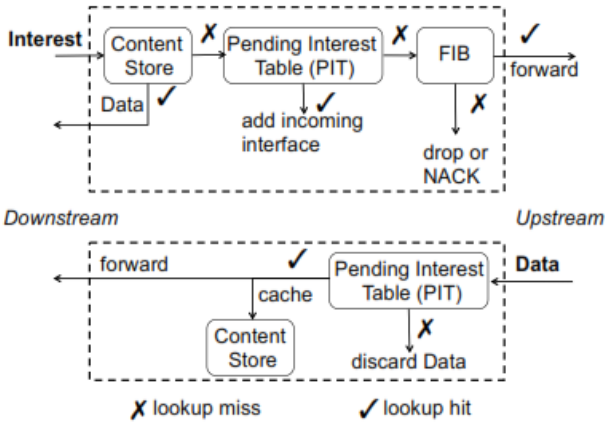
Terdapat tiga bagian utama dari *node* NDN, dapat dilihat pada Gambar II.2, dilihat dari perbedaan fungsinya (i) *Content Store* (CS) bagian dimana konten yang telah dikirimkan untuk *request* sebelumnya disimpan untuk sementara waktu; (ii)

Pending Interest Table (PIT), rekaman/catatan dari *interest* yang telah dilayani oleh *router*, namun konten yang diminta belum terkirim; (iii) *Forward Information Base* (FIB), rekaman/catatan terkait jalur posisi keberadaan konten di jaringan.



Gambar II.2 *Router* NDN

Adapun mekanisme *forwarding* pada NDN dapat dijabarkan sebagai berikut, dapat dilihat pada Gambar II.3. Pertama, consumer mengirimkan paket *interest* untuk meminta sebuah konten. *Router* membaca name prefix dan mencari konten yg diminta pada CS. Jika ada kecocokan, maka konten akan segera dikirimkan ke konsumer, jika tidak, maka paket *interest* akan diteruskan ke PIT. Kemudian PIT memeriksa apakah ada rekaman/catatan untuk permintaan yang sama yang telah dipesan. Jika ada, PIT akan menambahkan *interface* konsumer kedalam catatan *interest* yang pending, jika tidak maka catatan baru akan dibuat dan juga meneruskan paket *interest* ke FIB untuk mencari jalur dimana konten berada (*node* lain atau produser).



Gambar II.3 Mekanisme *forwarding* NDN (Zhang et al., 2014)

II.2 Perutean NDN

Mekanisme perutean NDN cukup berbeda dari mekanisme perutean pada jaringan IP. Sebelum melangkah lebih jauh maka ada baiknya kita lihat lagi dasar terkait pengiriman paket. Secara umu dalam memastikan sebuah paket dapat terkirim dari satu *node* ke *node* lain pada jaringan maka dibutuhkan paling tidak dua mekanisme yaitu perutean dan *forwarding*. *Rouitng* adalah mekanisme pencarian jalur terbaik bagi tujuan paket. *Forwarding* adalah mekanisme meneruskan paket ke jalur terbaik yang sudah di tentukan oleh mekanisme perutean.

Yi, (Yi et al., 2013) menggambarkan perbedaan antara *forwarding* dan perutean pada jaringan IP terhadap jaringan NDN sebagaimana dapat dilihat pada Tabel II.1. Pada mekanisme perutean NDN dan IP sama-sama menjalankan mekanisme perutean *stateful*, dimana sama-sama menggunakan algoritma pencarian jalur terbaik untuk menentukan jalur mana yang nanti dipilih sebagai *best route*. Dikatakan *stateful* karena algoritma berjalan *smart* dan adaptif dapat menyesuaikan dengan perubahan-perubahan yang terjadi di jaringan. Jalur yang merupakan perhitungan sebagai jalur terbaik hasil dari mekanisme perutean kemudian direkam/dicatat sebagai log database yang akan digunakan untuk proses *forwarding*.

Tabel II.1 Perbandingan Perutean dan *Forwarding*.

Tipe Jaringan	Tipe *Routing*	Tipe *Forwarding*
Jaringan NDN	*Stateful*	*Stateful*
Jaringan IP	*Stateful*	*Stateless*

Pada mekanisme *forwarding*, jaringan IP menjalankan mekanisme *stateless*, juga disebut *dumb forwarding*, dimana mekanisme yang terjadi hanya melihat log database rute yang telah dibuat mekanisme perutean dan meneruskan paket sesuai log rute tersebut. Sedangkan pada jaringan NDN menjalankan mekanisme *stateful* karena tidak hanya melihat log database rute dan meneruskan paket, *forwarding* juga mencatat dan mengukur performa rute selama pengiriman. Sehingga jika terjadi perubahan performansi jalur yang semual dipilih (link putus / penurunaan performa karena peningkatan utilitas jalur) *forwarding* dapat mengupdate log rute

pada database dengan jalur yang lebih baik. Sehingga yang amat jauh berbeda dari jaringan NDN dibandingkan jaringan IP adalah mekanisme *forwarding*nya, dimana mekanisme *forwarding* NDN lebih "pintar" dibandingkan dengan IP. Hal ini juga menimbulkan perdebatan apakah mekanisme perutean masih dibutuhkan di jaringan NDN

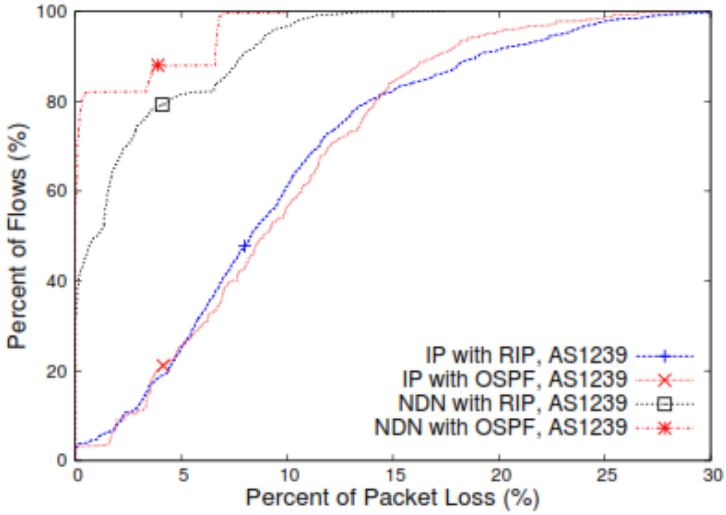
Yi, (Yi et al., 2013) menjelaskan pada penelitiannya bahwa mekanisme *forwarding* sendirian, tanpa melibatkan mekanisme perutean, pada *router* NDN dapat menangani kegagalan link secara efektif. Hal ini dikarenakan strategi *forwarding* pada jaringan NDN memiliki fitur untuk menangani kegagalan link dimana hal ini biasanya ditangani oleh mekanisme perutean seperti pada jaringan IP. Hal ini mengundang pertanyakan "apakah NDN *router* masih membutuhkan mekanisme perutean?". Namun disayangkan penelitian ini tidak menjabarkan bagaimana mekanisme perutean pada waktu awal terbentuk.

Yi, (Yi et al., 2014) memperbaiki penelitian sebelumnya dimana juga menjawab pertanyaan tentang kebutuhan keberadaan mekanisme perutean. Dimana pada penelitian ini ditunjukkan bahwa mekanisme *forwarding* telah gagal dalam (i) menentukan peringkat interface pada fase inisial (ii) mengembalikan log link yang telah recovery dan sebelumnya telah dihapus dari database.

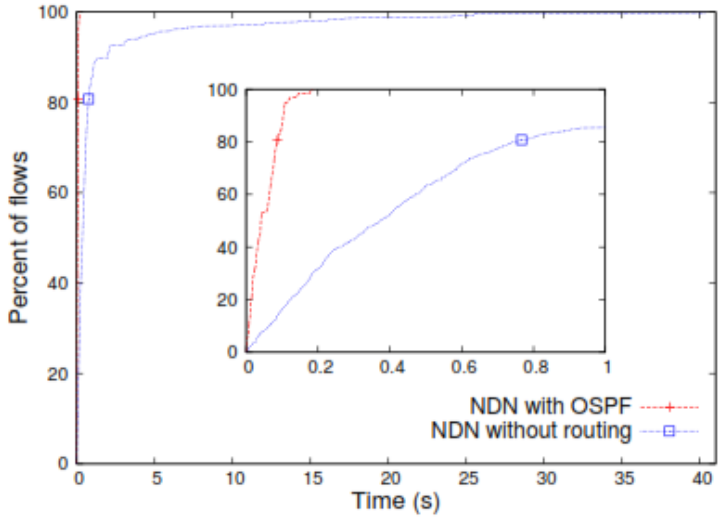
Pada penelitian ini (Yi et al., 2014) menguji coba pengaruh perutean pada jaringan NDN. Melihat pengaruh protokol perutean RIP dan OSPF pada jaringan IP dan NDN. Menguji coba pengaruh jaringan NDN jika menggunakan protokol perutean dan hanya mengandalkan kemampuan *probe* dari mekanisme *forwarding*.

Pada percobaan menggunakan protokol RIP dan OSPF pada jaringan IP dan NDN, dapat dilihat pada Gambar II.4. NDN dengan RIP (*distance vector*) terlihat mengungguli IP dengan RIP dan OSPF (link state). *Distance vector* dikenal dengan waktu konvergensi yang lambat, namun NDN memiliki kemampuan *stateful forwarding* yang dapat membantu meningkatkan kemampuan konvergensi rute. *Link state* pada NDN paling superior di antara ketiga percobaan, karena OSPF

memiliki kemampuan yang lebih dalam memetakan *cost* link di jaringan, sehingga pemeringkatan pilihan jalur alternatif dapat lebih baik.



Gambar II.4 Protokol Perutean RIP dan OSPF Pada Jaringan IP dan NDN (Yi et al., 2014)

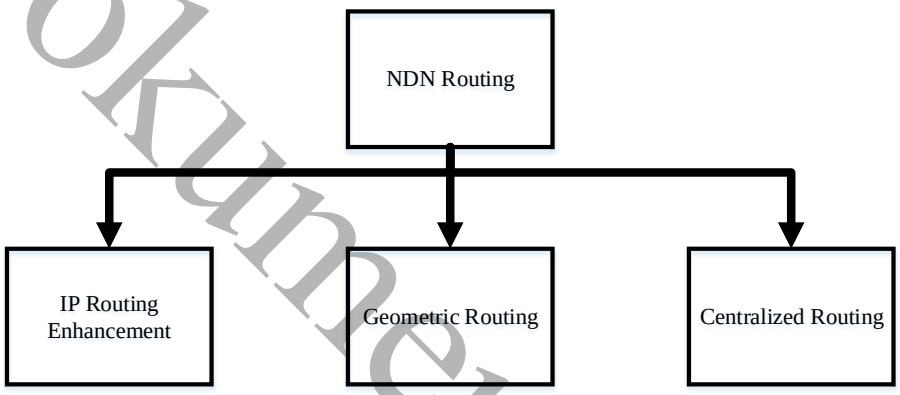


Gambar II.5 NDN Dengan dan Tanpa Protokol Perutean (Yi et al., 2014)

Percobaan dengan dan tanpa protokol perutean pada NDN, sebagaimana ditunjukkan pada Gambar II.5, memberikan pembuktian bahwa protokol perutean masih dibutuhkan pada jaringan NDN. Dapat dilihat dari persentase *flow* yang dapat diteruskan berdasarkan waktu. Seiring jumlah *node* yang bertambah perbedaan performansi dengan dan tanpa protokol perutean akan semakin signifikan.

Yi, (Yi et al., 2014) telah membuktikan bahwa perutean protokol memiliki peranan yang tidak tergantikan oleh mekanisme *forwarding*, setidaknya dalam (i) menentukan peringkat interface pada fase inisial; (ii) mengembalikan log link yang telah recovery dan sebelumnya telah dihapus dari database.

Berdasarkan penelitian yang telah ada maka mekanisme perutean NDN dapat diklasifikasikan secara umum menjadi tiga mekanisme perutean yang utama, sebagaimana dapat dilihat pada gambar II.6. dan akan dijelaskan pada bagian selanjutnya.



Gambar II.6 Klasifikasi Mekanisme Perutean NDN

II.2.1 Peningkatan Perutean IP

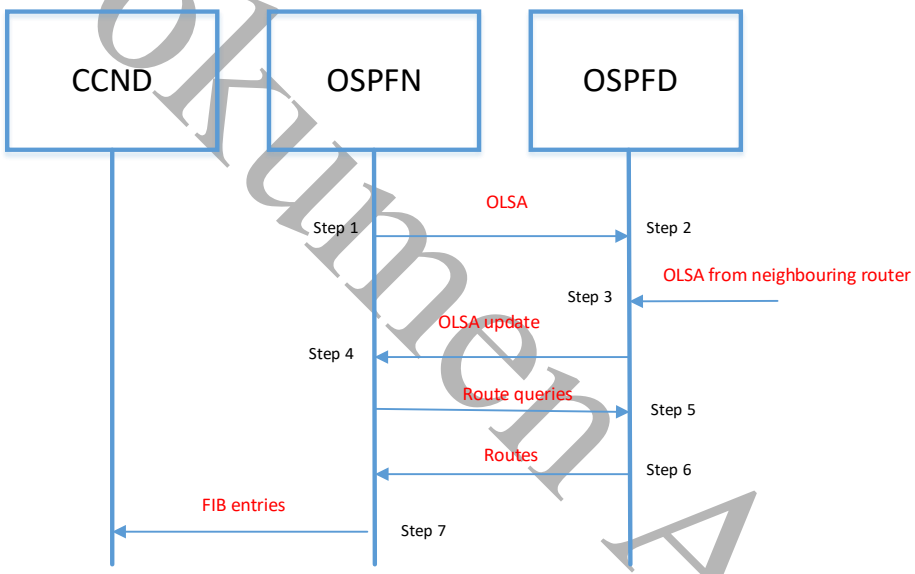
Pada kelompok ini mekanisme perutean untuk NDN dibangun dan dikembangkan langsung dari perutean protokol IP-based.

1) OSPFN

Mekanisme perutean NDN yang pertama kali diusulkan adalah OSPF for Named-data (OSPFN) (Wang and Hoque, 2012), merupakan pengembangan dari protokol perutean IP-based yang telah digunakan secara luas Open Shortest Path First (OSPF)(Moy, 1998). OSPFN dikembangkan sesuai dengan deskripsi dari Van Jacobson (Jacobson et al., 2009), protokol perutean pada jaringan IP akan dapat bekerja untuk jaringan NDN networks. OSPFN yang pertama kali dikembangkan adalah OSPFN versi 1.0, dimana telah dapat mengakomodir name-based prefix.

OSPFN versi 1.0 hanya dapat mendukung perutean single-path. Kemudian dikembangkan OSPFN versi 2.0 yang memiliki kemampuan yang penting dalma mendukung multipath terkonfigurasi. Diharapkan pengembangan pada OSPFN versi 3.0 untuk adaptif multipath, namun belum ada konfirmasi terhadap hal tersebut (Wang and Hoque, 2012).

Protokol perutean OSPFN didesian menggunakan tiga bagian utama yang bertanggung jawab menangani perutean based on content-centric: (i) OSPFN, (ii) OSPF Daemon (OSPFD), dan (iii) Content-Centric Networking Daemon (CCND). Mekanisme interaksi antar bagian dapat dilihat pada gambar II.7.



Gambar II.7 Langkah Pertukaran Pesan Antara OSPFN, OSPFD, dan CCND (Saxena and Roorkee, 2016; Wang and Hoque, 2012)

Secara umum, mekanisme perutean OSPFN memiliki tujuh langkah utama (Wang and Hoque, 2012), (Saxena and Roorkee, 2016) sebagaiman ditunjukan pada Gambar II.7. Langkah 1, ketika *node* hidup pertama kali, *node* akan mengalokasikan OLSA (Berger, 2008) untuk setiap name prefix yang akan diberitahukan kepada jaringan. Langkah 2, OSPFN meneruskan OLSA yang telah dialokasikan kepada OSPFD yang bertugas menyebarkan ke seluruh jaringan dengan mekanisme *flooding*. Langkah 3, ketika OSPFD menerima OLSA dari *node* lain, OSPFD memberikan dan mengirimkan update OLSA kepada OSPFN.

Langkah 4, jika OLSA yang diterima merupakan yang paling *update*, OSPFN meng-ekstrak nama *prefix* dengan *router*-ID dari database OLSA dan mencatatnya kedalam tabel name prefix. Langkah 5, OSPFN mengirimkan permintaan rute kepada OSPFD untuk mencari rute setiap nama *prefix*. Langkah 6, OSPFD menerima pesan permintan dari OSPFN, OSPFD memeriksa tabel perutean untuk jalur yang diminta dan mengirimkan jawaban kepada OSPFN. Langkah 7, OSPFN mengirimkan FIB *entries* ke CCND yang berisi nama *prefix*, next-hop(s), dan path cost) untuk melakukan *update* CCND.

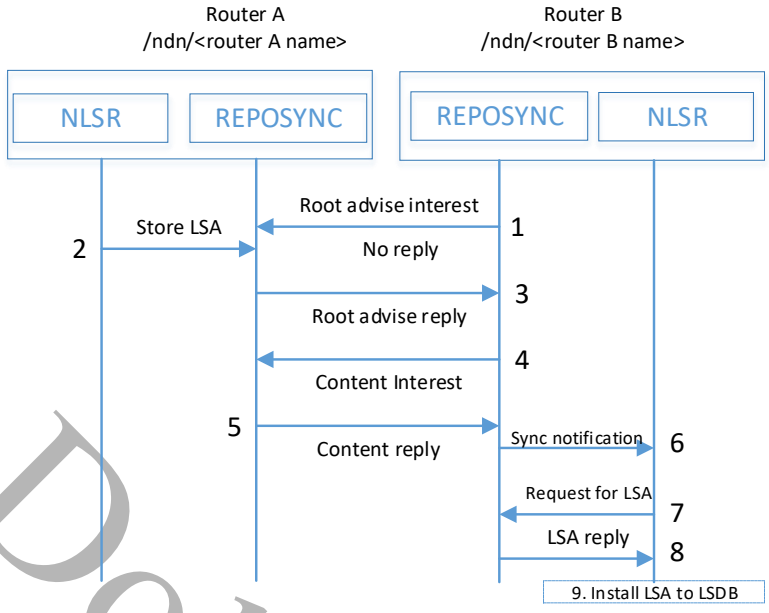
OSPFN membangun rute menggunakan OLSA advertise dengan mekanisme *flooding* untuk seluruh *node* di jaringan. Dalam mendukung kompatibilitas dengan jaringan IP, *node* IP-based juga melakukan *forwarding* OLSA tanpa menggunakan OLSA untuk membangun topologi jaringan pada dirinya.

Oleh karena OSPFN dibangun berdasarkan standar OSPF, maka OSPFN masih menggunakan alamat IP sebagai *Router*-ID dan bergantung pada tunnelling Generic *Routing* Encapsulation (GRE). Mekanisme yang dobel antara IP, pengalamatan dan tunneling, serta name prefix meningkatkan *overhead* pada mekanisme perutean yang secara tidak langsung membatasi kemungkinan menciptakan *forwarding* multi-path (Saxena and Roorkee, 2016).

2) NLSR (Hoque et al., 2013; Lehman et al., 2017)

Dikarenakan masih sangat bergantung pada IP, OSPFN, memberikan tantangan pada usulan *Named-data Link State Routing* (NLSR) sebagai protokol perutean pertama yang terdistribusi berdasarkan NDN. NLSR didesain bekerja pada jaringan NDN dan melepaskan ketergantungan pada mekanisme IP.

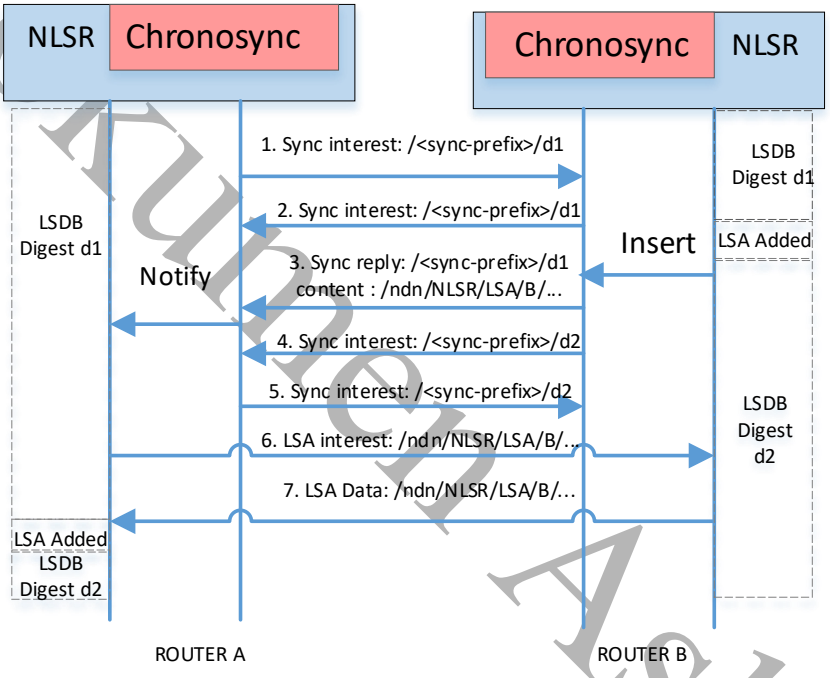
Hoque, et al. (Lehman et al., 2017) mengusulkan empat kunci utama pada desain tersebut (1) menggunakan skema nama bertingkat untuk *router*, keys, dan update perutean, (2) model trust bertingkat, (3) protokol sinkronisasi hop-demi-hop, dan (4) memberikan peringkat banyak rute untuk pilihan *forwarding*.



Gambar II.8 Mekanisme NLSR Menggunakan CCNx Sync/Repo (Hoque et al., 2013)

Gambar II.8, menunjukan bagaimana LSA dari mekanisme NLSR disebarkan pada jaringan untuk menyusun mekanisme perutean, Hoque, et al. (Hoque et al., 2013) menjelaskan mekanisme tersebut sebagai berikut. *Router* A dan B secara periodik melakukan sinkronisasi terhadap bagian network yang berisi LSA. Pada kasus ini, *router* B mengirimkan pesan *Interest* khusus, dinamakan *Root Advise*, berisi kombinasi nilai hash dari *name-prefix* kepada *node* tetangga (langkah 1). Ketika NLSR *Router* A membuat LSA, ia juga mencatatnya pada bagian *Sync* (langkah 2), bagian *Sync Router* A membandingkan nilai hash dengan nilai hash yang berasal dari *Sync Router* B's, dan jika berbeda, maka *Sync Router* A mengirimkan *Root Advise reply* ke *Sync Router* B dengan nilai hash dari record semula (langkah 3). Bagian *Sync Router* B kemudian membandingkan nilai hash dan memeriksa level berikutnya dari nilai hash mana yang berbeda. Dari proses ini, *Sync Router* B akan menemukan LSA yang yang harus disinkronisasi dan dilanjutkan dengan mengirimkan permintaan sinkronisasi dengan menggunakan pesan Interest (langkah 4 dan 5). Setelah sinkronisasi selesai *Sync Router* B kemudian mengirimkan *Sync* Notifikasi kepada NLSR *Router* B (step 6), yang akan menarik data dari repo *Router* B (langkah 7 dan 8) dan melakukan *update* pada LSDB (langkah 9).

Repo/*Sync* pada NLSR diimplementasikan menggunakan CCNSync yang merupakan bagian dari CCNx Repo ("CCN web," n.d.). Data yang diterima oleh CCNSync juga dicatat di dalam Repo dan catatan ini secara default tidak dapat dihapus. Mekanisme ini dapat menyebabkan masalah pada sumber daya memory pada sistem NLSR setelah berjalan beberapa saat. Lehman, et al. (Lehman and Wang, 2016) mengusulkan peningkatan dengan mengganti CCNSync dengan ChronoSync (Zhu and Afanasyev, 2013). Perubahan ini, juga memiliki konsekuensi perubahan mekanisme, seperti dapat dilihat pada Gambar II.8.



Gambar II.9 Mekanisme NLSR Menggunakan ChronoSync (Lehman and Wang, 2016)

Gambar II.9 menunjukan bagaimana LSA disebarkan pada jaringan dengan perubahan sistem menggunakan ChronoSync (Zhu and Afanasyev, 2013). ChronoSync bertanggungjawab dalam melakukan sinkronisasi LSA pada LSDB. Untuk tujuan tersebut, protokol ChronoSync mengirimkan Sync Interests, berisi nilai hash name prefix LSA, kepada *node* di jaringan (langkah 1 dan 2). Ketika LSA baru ditambahkan pada LSDB *Router* B, kemudian LSA set pada *Router* akan terupdate. ChronoSync membalas dengan cara mengirimkan *Sync Reply* kepada

Router A dengan LSA berisi nama yang baru (langkah 3). ChronoSync *Router* A menerima *Sync Reply* data berisi LSA yang baru, kemudian memberitahukan NLSR *Router* A terkait perbedaan data, dan mengupdate set LSA berisi nama yang baru. *Router* A dan B kemudian menghitung nilai hash yang baru untuk set nama tersebut dan mengirimkan Sync Interest yang baru berisi hash update (langkah 4 dan 5). Karena proses NLSR pada *Router* A telah diberi tahu diberitahu terkait hilangnya data LSA, NLSR mengirimkan LSA Interest untuk menarik data untuk LSA yang hilang (langkah 6). *Router* B memberikan jawaban pada Interest tersebut terkait permintaan data LSA (langkah 7). Ketika NLSR *Router* A menerima data LSA, kemudian data LSA tersebut disisipkan kedalam LSDB. Setelah proses tersebut, maka LSDB kedua *router* tersebut telah sinkron.

ChronoSync memperbaharui mekanisme NLSR dalam megupdate data LSA terkait nama prefix, kemudian NLSR dapat menarik data tersebut dan memperbaharui LSDB dengan hanya melihat versi terakhir dari LSA, dan LSA versi sebelumnya akan dihapus. Sebagai hasilnya NLSR dapat mengatasi masalah memori yang terjadi pada versi NLSR sebelumnya. ChronoSync juga menggunakan mekanisme broadcast untuk melakukan sinkronisasi diantara *router* berbasis NLSR dimana akan menghasilkan konvergensi route yang lebih cepat, namun berkontribusi meningkatkan permasalahan perutean *overhead* yang berdampak pada skalabilitas protokol perutean.

II.2.2 Perutean Geografis

Implementasi NDN pada jumlah perangkat yang masif seperti pada Internet of Things (IoT) dan Wireless Sensor Network (WSN) secara natural, akan berhadapan dengan masalah jumlah pesan broadcast (*routing overhead*) dan keterbatasan sumberdaya pada *node*, lebih jauh lagi, implementasinya juga akan berhadapan dengan jumlah database yang sangat besar, e.g., *forwarding tables*, *pending interest*, dll., pada masing-masing *node*. Untuk mengatasi masalah tersebut, perutean geometric (Karp and Kung, 2000), (Kuhn et al., 2003) digunakan untuk memberikan koordinat/lokasi untuk masing-masing *node*, berdasarkan referensi tertentu, e.g., koordinat bumi, popularitas, dll. Kemudian masing-masing *node*

menentukan, biasanya, jalur terpendek menuju *node* tujuan melalu *node* tetangga. Karena telah dibuktikan bahwa untuk mencari jalur terbaik dari hubungan sumber dan tujuan hanya dibutuhkan informasi *node* tetangga terdekat maka semua tantangan terkait *message overhead* dan sumber daya tadi dapat di hemat (Karp and Kung, 2000).

Teknik perutean geografis yang paling banyak dan efisien digunakan dalam menentukan jalur terpendek adalah algoritma greedy algorithm, dengan teknik ini semua *node* akan memforward pesan kepada *node* tetangga yang memiliki lintasan terpendek menuju *node* tujuan. Pada penelitian (Karp and Kung, 2000), (Kuhn et al., 2003), telah ditunjukkan mekanisme greedy, dengan asumsi bahwa semua *node* mengetahui posisi mereka dengan acuan referensi tertentu seperti dimensi ruang fisik. Hasilnya menunjukan perutean geografis berbasis greedy memiliki efisiensi pada perutean *overhead* namun kenyataannya algoritma tersebut tidak sepenuhnya menjamin terbentuknya jalur dari sumber ke tujuan. Karena pada kasus tertentu, pesan bisa saja terjebak di sebuah situasi yang disebut “local minima” (Gao, 2009). Kondisi tersebut biasanya dinyatakan disebabkan oleh “lakes” atau “voids”, terjadi ketika muncul ruang kosong yang besar di antara *node* pada jaringan.

Pengembangannya Kleinberg (Kleinberg, 2007) mengusulkan dan membuktikan peta jaringan berdasarkan koordinat hiperbolik dengan mekanisme greedy selalu lebih baik dalam mendistribusikan trafik dan mengatasi kongesti pada *node-node* yang populer.

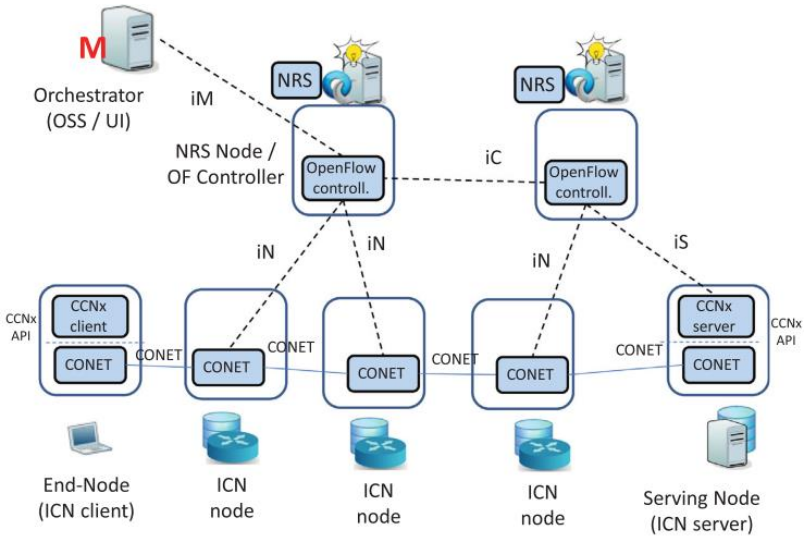
NDN menggunakan perutean hiperbolik (RH) telah diteliti pada (Lehman et al., 2016). RH memanfaatkan algoritma greedy untuk melakukan pencarian jalur pada koordinat hiperbolik dari *node* di jaringan (Boguñá et al., 2010), (Krioukov et al., 2009). Dengan syarat, masing-masing *node* memiliki informasi terkait koordinat posisinya, koordinat tetangganya dan masing-masing paket Interest membawa informasi koordinat sesuai dengan name-prefix konten yang diminta, kemudian mekanisme forwading menghitung jarak *node-node* tetangga dari informasi koordinat yang ada ke next-hop terbaik, dengan kata lain memiliki jarak terpendek.

Pada penelitian (Lehman et al., 2016), mengusulkan RH sebagai bagian mekanisme perutean NDN, karena RH memiliki kelebihan, (i) secara real-time dapat menentukan dan menghitung jalur terbaik, dan juga minimal dalam penggunaan sumber daya, memori dan perutean overead. (ii) diharapkan dapat stabil menghadapi perubahan topologi, i.e., kemampuan backtracking. (iii) RH lebih dapat diandalkan dibandingkan perutean geografis murni (Aboud and Touati, 2016) dalam mendistribusikan trafik secara merata pada jaringan. Walaupun namespace pada NDN jauh lebih besar dari alamat IP dengan mekanisme RH, didapat penghematan secara exponensial dari namespace dan *overhead*.

Perutean NLSR berbasis link-state telah berhasil dengan cukup baik dikembangkan dari perutean IP, OSPF. Namun algoritma link-state berkontribusi pada perutean *overhead* yang cukup besar karena menggunakan mekanisme broadcast dalam menyebarkan informasi rute ke seluruh *node*. Perutean berbasis geografis menawarkan *overhead* yang lebih rendah dapat dilihat dari usulan geografis perutean yang awal diusulkan B. Karp and H. T. Kung, *Greedy Perimeter Stateless Routing* (GPSR) (Karp and Kung, 2000).

II.2.3 Perutean ter-Sentralisasi

Beberapa tahun belakangan ini paradigma baru, jaringan tersentralisasi memiliki pengaruh yang sangat besar pada jaringan internet. Bergeser dari paradigma lama dimana jaringan memiliki kontrol yang terdistribusi, di masing-masing *node*, menjadi kontrol terpusat, dinamakan Software Defined Network (SDN) menggunakan protokol OpenFlow (McKeown et al., 2008; “ONF Web,” n.d.; “Opendaylight web,” n.d.). SDN merupakan desain arsitektur jaringan dimana memusatkan fungsi lapis kontrol dan memisahkan antara fungsi lapis kontrol dan user. *Node* berbasis SDN menyederhanakan fungsi dari *node* sebelumnya, dimana hanya bertanggung jawab pada fungsi data *forwarding*. Fungsi kontrol seperti perutean, dan mekanisme kebijakan ditangani oleh *node* pintar yang diberi nama controller.



Gambar II.10 ICN CONET-SDN Architecture (Salsano et al., 2013)

Dikarenakan mekanisme NDN melibatkan name dan konten, hal tersebut akan menghasilkan lebih banyak ID dibandingkan jaringan berbasis IP. Hal tersebut berhubungan juga dengan jumlah konten pasti akan jauh lebih banyak dibandingkan IP alamat host yang digunakan pada jaringan IP. Oleh karen itu, beban pada *node* dalam menjalankan perutean name-based, dalam mencari jalur antara konsumer dan konten yang diinginkan dapat menjadi sangat tinggi dalam konteks mekanisme pemrosesan.

Melazi, et al. (Salsano et al., 2013) mengusulkan desain jaringan berdasarkan ICN dan arsitektur SDN OpenFlow sebagaimana dapat dilihat pada Gambar II.10. Mereka mengusulkan untuk menggunakan CONET (Andrea Detti et al., 2011), (A Detti et al., 2012) sebagai kerangka kerja penamaan konten dan membangun *testbed* pada proyek OFELIA (“Ofelia Project,” n.d.) di Eropa. Penelitian tersebut menggunakan protokol CONET dan CCNx dalam membangun *node* berdasarkan kontena namun tetap masih menggunakan beberapa mekanisme IP based. Arsitektur ini membedakan lapis dari operasi menjadi 2 (dua) bagian (i) data plane berhubungan dengan *server* ICN *Server* (sebagai produser konten), klien ICN (sebagai konsumer) dan *node* ICN (sebagai intermediate *node*) dan (ii) control plane berhubungan dengan *node Name Routing System* (NRS), infrastruktur keamanan,

e.g., PKI – Public Key Infrastructure, dan ‘‘Orchestrator’’ *node*s, yang bertanggung jawab dalam management dan orchestration jaringan (Salsano et al., 2013).

Pada penelitian (Charpinel et al., 2016)–(Torres et al., 2012) juga mengusulkan skema berbasis SDN untuk berkolaborasi dengan NDN. Fungsi SDN menawarkan skalabilitas yang lebih baik dengan membedakan data plane dan control plane. Lebih jauh lagi, dengan fungsi control plane yang tersentralisasi memungkinkan mekanisme yang kompleks seperti smart *forwarding* dan smart caching menjadi lebih mudah diimplementasikan tanpa memberikan lebih banyak beban pada perutean *overhead*. Sebaliknya penggunaan controller yang terpusat mensaratkan kehati-hatian dalam keputusan penempatan *node* controller dan penanganan terkait masalah keamanan dll, karena controller merupakan single point of failure pada arsitektur SDN.

II.3 Skalabilitas NDN

Problem Skalabilitas pada masalah perutean adalah terkait bagaimana mekanisme perutean dapat tetap berjalan baik seiring bertambahnya ukuran/skala jaringan, atau secara sederhana, perutean dapat menangani puluhan, ratusan, ribuan, atau ratusan juta *node* pada jaringan NDN. Tentu saja permasalahan akan semakin kompleks berkali lipat untuk jaringan yang amat besar. Berdasarkan mekanisme yang ada di NDN paling tidak ada 3 hal yang akan berpengaruh pada skalabilitas perutean NDN yaitu: (i) efisiensi penamaan, (ii) jumlah record perutean tabel (iii) paket *overhead.*

Salah satu keuntungan dari penggunaan NDN sebagai teknologi IP adalah penggunaan nama prefix berbentuk Uniform Resource Identifier (URI). Dengan penggunaan URI paling tidak memilki keuntungan (i) network dapat mengetahui keinginan konsumer secara eksplisit, (ii) penggunaan URI tidak dibatasi dan tidak terbatas. Sementara itu Yuan, dkk (Yuan et al., 2012) membuktikan bahwa panjang dari URI memiliki tradeoff terhadap performa *node* dalam melakukan *forwarding* data. Pada penelitian tersebut peningkatan segmen URI, dari 2 segmen menjadi 32 segmen, menurunkan performa troughput *node* NDN hingga sekitar 33%. Terkait hal tersebut skema nama yang efisien akan terkait erat dengan skalabilitas NDN.

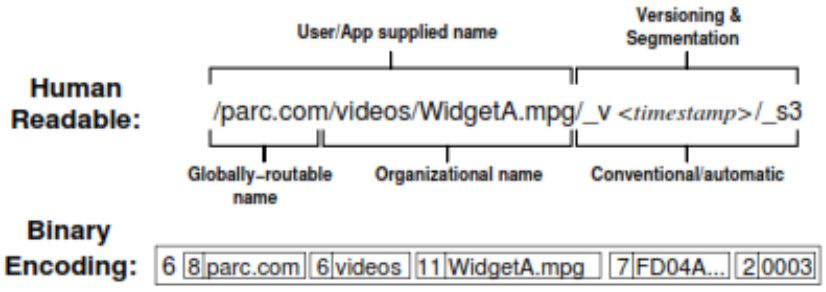
Seiring perkembangan internet dengan pertumbuhan jumlah user, domain, jenis servis dan berbagai teknik modifikasi jaringan menyebabkan pada domain perutean atau pencarian jalur menjadi sangat kompleks. Sebagai gambaran, pada *router Default Free Zone* (DFZ), *router* dengan tabel perutean trebesar pada BGP, memiliki 7x105 rute alamat("BGP Reports," n.d.). Lebih jauh untuk menggambarkan skalabilitas internet yang harus dihadapi perutean pada NDN, adalah jumlah website di internet pada tahun ini adalah 18x109 ("Total number of Websites - Internet Live Stats," n.d.) dan jumlah total halaman web yang disediakan oleh situs web berjumlah 48x109 halaman("WorldWideWebSize.com | The size of the World Wide Web (The Internet)," n.d.). Sementara data jumlah halaman web tersebut adalah total halaman web yang telah diindex oleh mesin pencari google sampai 2018, dan belum seluruhnya menggambarkan seluruh konten yang tersebar di internet. Google tentu saja tidak melakukan index terhadap konten file sharing atau yang bersifat lokal, servis seperti peer to peer, dan halaman web yang memang sengaja disembunyikan. Jika asumsi besaran konten 1,5 – 2 x jumlah halaman web atau sekitar 1011 konten maka jumlah tersebutlah yang harus ditangani oleh *router* NDN. Yang artinya jika dibandingkan dengan perutean BGP, 105 lebih banyak dari perutean table BGP pada *router* DFZ. Dari data tersebut sangatlah krusial penanganan skalabilitas perutean pada NDN.

Penelitian pada NDN juga telah bergerak kearah *node* NDN dengan kemampuan mobilitas. Tentu hal ini berhubungan erat dengan karakteristik dari jaringan tanpa kabel, Internet of Things (IoT), ad-hoc, dll yang memiliki karakter tersebut. NDN sebagai teknologi pengganti IP tentu harus dapat menangani tantangan yang ada pada skema jaringan tersebut. *Node* yang mobile menyebabkan mekanisme perutean harus dapat beradaptasi terhadap perubahan tersebut yang tentu saja berimplikasi terhadap pilihan jalur terbaik dalam pengiriman data. Pada dasarnya mekanisme perutean pada IP seperti berbasis link state atau distance vector tidak diciptakan untuk beradaptasi pada lingkungan mobile. Seingga perutean pada NDN yang menggunakan mekanisme yang sama seperti NLSR (Lehman and Wang, 2016) akan memiliki karakter yang sama. Lehman, dkk (Lehman et al., 2016) telah

memperlihatkan jumlah *node* akan meningkatkan *overhead* perutean secara linier hingga 335 link, dan selebihnya, peningkatan *overhead*, mendekati eksponensial.

II.4 Penamaan

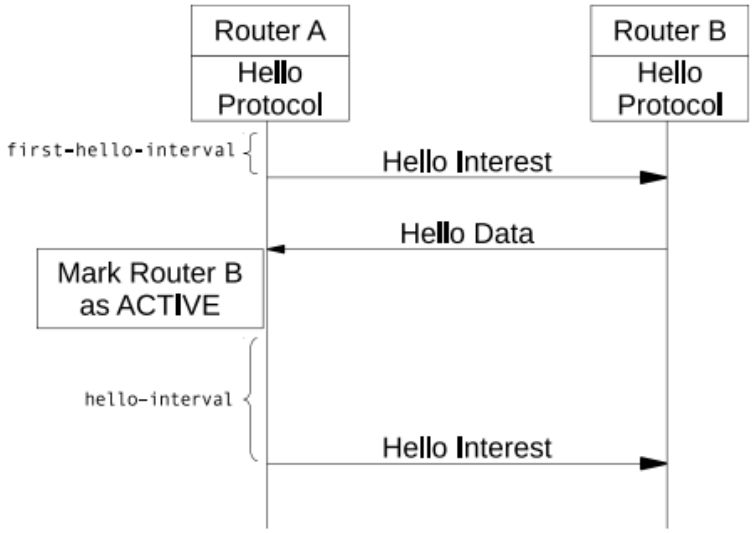
Penamaan pada NDN merupakan struktur yang memiliki hierarki, sehingga sebuah nama terdiri dari sejumlah komponen yang menyusun hierarki tersebut dan tidak ada aturan terhadap jumlah komponen yang digunakan. Setiap komponen dari nama dikodekan kedalam komponen oktet atau biner yang dapat berubah-ubah panjangnya. Nama harus memiliki arti untuk layer atas NDN Protocol Stack supaya dapat digunakan untuk pengiriman data. Pengkodean langsung secara biner atau metode lain yang lebih kompleks dapat digunakan secara langsung tanpa harus melakukan perubahan kembali kedalam teks selama transmisi. Dan secara umum demi kesesuaian dengan proses yang ada dan notasi yang sederhana maka Jacobson (Jacobson et al., 2009) mengusulkan bahwa penamaan memiliki struktur menyerupai URI dengan karakter "/" sebagai pemisah antar komponen, sebagaimana dapat dilihat pada Gambar II.11. Pada contoh ini digambarkan kesepakatan penamaan pada level aplikasi yang digunakan untuk melakukan pembedaan terhadap perubahan file yang sama sepanjang waktu (info versi dilambangkan dengan, _v dikodekan sebagai FD, diikuti oleh nomor versi integer) dan pembagian file (info segmen dilambangkan dengan, _s dikodekan sbagai 00, diikuti oleh nilai integer yang dapat berarti blok atau nomor byte number atau nomor frame dari video frame pertama pada paket). Komponen terakhir dari setiap paket data dapat menggunakan SHA256 digest sebagai info keseluruhan paket.



Gambar II.11 Contoh Penamaan NDN (Jacobson et al., 2009)

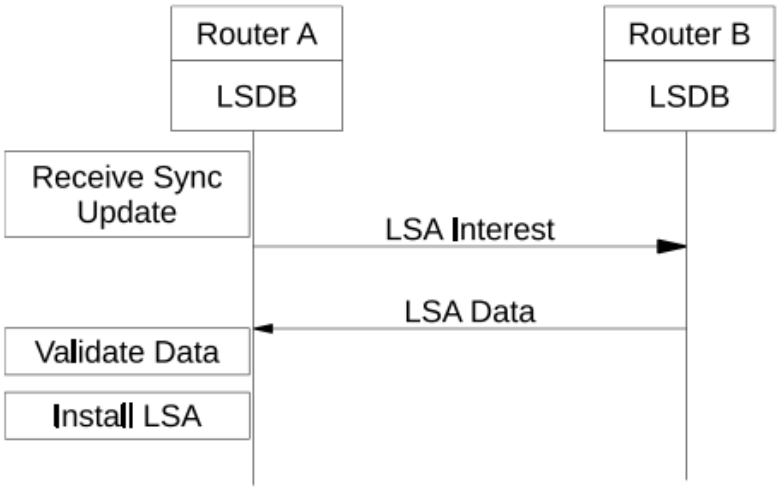
II.5 Tabel Perutean dan *Forwarding*

Dalam rangka melakukan proses pemilihan jalur terbaik pada jaringan maka setiap *node* harus melakukan pemetaan terhadap seluruh kemungkinan jalur yang ada di jaringan. Proses pertama yang dilakukan dan akan terus dilakukan selama *node* aktif adalah melakukan cek terhadap keberadaan tetangganya. Secara umum proses ini dilakukan dengan mengirimkan pesan hello secara broadcast ke seluruh interface yang terhubung pada *node*, sebagaimana dapat dilihat pada Gambar II.12. Ketika paket hello yang dikirimkan oleh *node* A mendapatkan balasan dari *node* B, maka *node* A akan mencatat pada tabel route bahwa *node* B aktif. Hal ini akan dilakukan secara periodik dalam selang waktu tertentu



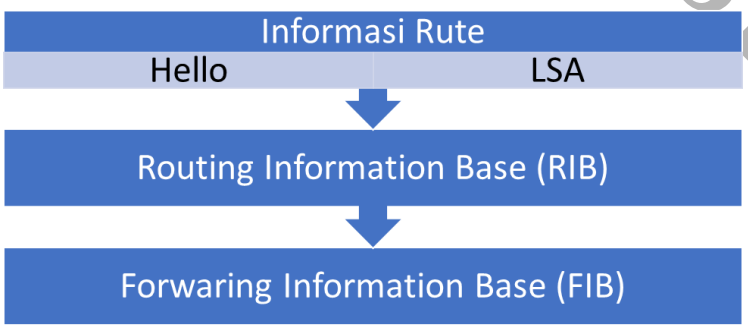
Gambar II.12 Pengiriman pesan Hello

Setelah inisialisasi pada *node* tetangga selesai maka *node* NDN akan melakukan pertukaran database data (LSA) yang berisi peta jalur jaringan dilihat dari sudut pandang tetangganya, sebagaimana dapat dilihat pada Gambar II.13. *Node* A akan mengirimkan permintaan LSA (LSA Interest) kepada *node* B, kemudian *node* B akan membalas dengan memberikan LSA data kepada *Node* A. *Node* A akan mencek validitas data LSA tersebut dan mengupdate record data yang ada padanya



Gambar II.13 Pertukaran informasi peta

Data dari mekanisme pesan Hello dan LSA akan dikompulir pada pusat recor bernama *Router Information Base* (RIB). Sehingga RIB akan berisikan seluruh kemungkinan rute dengan seluruh metric dan cost untuk mencapai rute tersebut. Kemudian *node* akan melakukan komputasi untuk memilih jalur mana yang terbaik untuk seluruh kemungkinan tujuan dan hasil pemilihan tersebut akan di simpan pada *Forward Information Base* (FIB), dapat dilihat pada Gambar II.14. FIB akan digunakan sebagai referensi ketika sebuah data yang masuk ke *node* ingin pergi ke sebuah tujuan tertentu. FIB berisi rekaman pilihan jalur terbaik kemana paket tersebut akan di teruskan. Jumalah record yang disimpan pada RIB dan FIB akan bertambah seiring pertambahan jumlah *node* dan konten yang ditransmisikan di jaringan.



Gambar II.14 Pembentukan FIB

II.6 *Overhead* Perutean

Mekanisme mendapatkan rute sebagaimana dijelaskan sebelumnya yaitu, pesan Hello dan data LSA dilakukan secara periodik dalam interval waktu tertentu. Hal ini dilakukan dalam rangka memperbaharui data final yang ada di FIB. Jika data FIB tidak terupdate dengan baik maka mekanisme penerusan pesan ke *node* selanjutnya akan terganggu.

Tabel II.2 Perutean NDN dan Skalabilitasnya.

No	Mekanisme Perutean	Referensi Utama	Mekanisme	Skalabilitas
1.	OSPFN	(Berger, 2008; Moy, 1998; Wang and Hoque, 2012)	- Peningkatan kemampuan OSPF - OLSA - Menggunakan REPO/SYNC - IP based	- Beban pada mekanisme IP - Repo menyebabkan masalah sumber daya memori - Skalabilitas sangat rendah
2.	NLSR	(Lehman et al., 2017), (Lehman and Wang, 2016)	- Perbaikan OSPFN - OLSA - Name Based - Menggunakan ChronoSync - Broadcast LSA	- Beban Route *overhead* - Ada perbaikan skalabilitas namun masih mewarisi mekanisme IP
3.	Perutean ter-Sentralisasi/ ICN CONET	(Salsano et al., 2013), (Charpinel et al., 2016; Ooka et al., 2013; Torres et al., 2012)	- SDN based - Centralized control plane	- Skalabilitas bukan lagi di *node* namun di kontroller
4.	Perutean Geografis /Hiperbolik	(Lehman et al., 2016), (Aboud and Touati, 2016)	- Location-based - Jalur terpendek - Low memory consumption	- Skalabilitas amat baik - Mekanisme bisa menjadi beban

Setiap pesan yang dikirmkan dalm rangka memperoleh informasi Hello dan LSA disebut perutean *overhead*. Atau dapat di definisikan setiap pesan atau data yang dikimkan dalam proses pencarian jalur atau perutean.

II.7 Klusterisasi

Berdasarkan (Afsar and Tayarani-N, 2014), maka teknik *clustering* pada jaringan memiliki tujuan dan fokus yang beragam antara lain:

a. Skalabilitas

Pada penelitian (Kleinrock and Kamoun, 1977), di jabarkan bahwa sebuah jaringan yang sangat luas dapat ditingkatkan skalabilitasnya dengan membagi layer-layer arsitektur jaringan ke dalam *cluster*. Ketika sebuah *node* ingin berkomunikasi dengan *node* lain di *cluster* yang berbeda, maka *Cluster* Head (CH) akan memfasilitasi komunikasi tersebut yang akan berakibat berkurangnya perutean table secara signifikan.

b. Fault-Tolerance

Dengan metode *clustering* maka gangguan pada sebuah *cluster* tidak akan mempengaruhi *cluster* yang lain. Beberapa teknik seperti pada (Tubaishat et al., 2004) menggunakan teknik *clustering* secara adaptif sehingga dapat mengupdate CH jika kesalahan terjadi

c. Agregasi data

Permintaan konten yang populer akan lebih kerap terjadi dibandingkan permintaan yang ditujukan pada konten yang tidak populer. Sehingga dengan teknik *clustering* dimungkinkan untuk di agregasi konten-konten yang populer pada *node* CH sehingga utilitas jaringan dapat lebih efisien

d. Load balancing

Penggunaan CH sebagai *node* koordinator akan secara cepat menguras energi pada *node* tersebut. Rotasi antara *node* yang paling “kuat” pada *cluster* dapat memeratakan penggunaan energi juga bisa menggunakan teknik load balancing dalam membagi beban pada CH

e. Stabilized network topology

Dikarenakan jaringan secara keseluruhan dibagi-bagi kedalam *cluster-cluster* yang lebih kecil, maka perubahan-perubahan yang terjadi dapat di lokalisir dan hanya berpengaruh pada *cluster* tersebut saja

f. Maximal network lifietimae (energi)

Kelebihan-kelebihan diatas dapat berujung pada penggunaan energi yang lebih efisien untuk keseluruhan *node*. Penggunaan energi yang lebih efisien akan memperpanjang usia jaringan atau network lifetime Sehingga metode *clustering* dapat diimplementasikan pada algoritma perutean NDN dalam meningkatkan skalabilitas

g. Hubungan sosial terhadap network

Jaringan pada NDN-IoT memiliki kedinamisan yang sangat tinggi, hal ini disebabkan variasi tipe *node* yang amat besar. *Node* pada NDN-IoT dapat saja berupa sebuah sensor pada sebuah lahan untuk mendeteksi parameter tanah sehingga sensor tersebut tidak bergerak, dapat pula berupa sensor tekanan darah dan jantung yang dipasang pada atlet lari dengan kecepatan sedang, dan bahkan juga dapat berupa sensor pada mobil untuk mengukur performa selama perjalanan dengan kecepatan tinggi. Oleh karena itu hubungan secara fisik antar *node* amatlah dinamis sehingga akan sering berubah dalam rangka memetakan rute pada jaringan

Hubungan dan interaksi sosial antar *node* merupakan parameter yang dikembangkan para peneliti untuk membangun keterkaitan antar *node* di jaringan agar lebih stabil. Penelitian menggunakan hubungan sosial pada jaringan cukup banyak digunakan pada Delay Tolerant Network (DTN) (Kaimin Wei et al., 2014). Pada algoritma dasar perutean DTN seperti spray-and-wait, maka ketika *node* akan mengirimkan pesan maka *node* tersebut akan membanjir jaringan dengan pessan tersebut dengan tujuan pesan tersebut sampai ke tujuan, atau biasa disebut mekanisme *flooding*. Dengan mekanisme *flooding* akan membuat jaringan terbebani dengan replikasi pesan yang amat banyak, menyebabkan *overhead* menjadi sangat tinggi. Dengan mekanisme hubungan sosial maka *node* dapat lebih selektif dalam memilih *node* kandidat untuk meneruskan pesan ke tujuan.

II.8 Model Mobilitas dan Pembangkitan Interest

Dalam memodelkan sistim NDN secara utuh, sebagai tahap awal perlu dibangun model mobilitas dan bagaimana sebuah interest dibangkitkan pada mekanisme pengiriman data.

II.8.1 Model Mobilitas

Pada kasus *node* NDN memiliki sifat mobile maka dapat menyebabkan perubahan status keterhubungan antar *node* (link), link aktif atau link non-aktif, pada sebuah periode pengamatan. Di modelkan bahwa sebuah vertex v berada pada jaringan NDN di sebuah area A, dimana area tersebut memiliki luasan lebih besar dibandingkan cakupan sinyal v. Kondisi edge e pada waktu tertentu t direpresentasikan sebagai variable acak e(t) = {0,1}, dimana bernilai 0 (nol) menyatakan link non-aktif, dan bernilai 1 untuk menyatakan link aktif. Sehingga dapat dijabarkan bahwa probabilitas status sebuah link berubah pada waktu pengamatan Plb (t) adalah sebagi berikut:

$$P_{lb}(t) \stackrel{\text{def}}{=} \begin{cases} P\{e(t) = 1 | e(t-\tau) = 0\} \\ P\{e(t) = 0 | e(t-\tau) = 1\} \end{cases} \quad \text{(II.1)}$$

dengan

$A = Area\ pengam$atan

$v = Node\ NDN$

$e = Link\ NDN$

$c = Cakupan\ node\ NDN$

$t = \ Waktu\ saat\ ini$

$t - \tau = \ Waktu\ sebelumnya$

Sementara distribusi dari lamanya penggunaan link telah banyak di usulkan dan dibuktikan sebagai distribusi eksponensial yang di verifikasi pada (Xu et al., 2007) dan (Liang Qin and Kunz, 2006) pada model mobilitas Random Walk dan semi-Markov smooth. Secara lebih spesifik, pada (Liang Qin and Kunz, 2006) mengusulkan bahwa durasi sebuah koneksi/link/edge dapat didekati dengan

distribusi eksponensial. Oleh karena itu pendekatan durasi link terbentuk dapat di dekati dengan

$$mean\ \mu_v = \frac{1}{\lambda_v} \quad \text{(II.2)}$$

sehingga CDF dan PDF dari durasi koneksi secara berurutan adalah:

$$P_{Clb}(t) = 1 - e^{-\lambda_v} \quad \text{(II.3)}$$

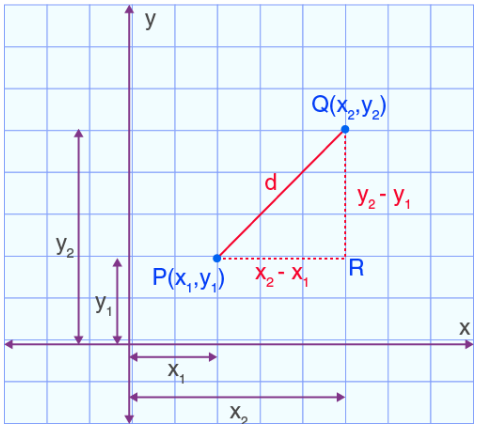
$$P_{Plb}(t) = \lambda_v e^{-\lambda_v} \quad \text{(II.4)}$$

dengan

$\mu_v = rate\ kedatangan\ node\ dan\ terbentuk\ koneksi$

$\lambda_v = rate\ kepergian\ node\ dan\ terputus\ koneksi$

Dalam memodelkan mobilitas para peneliti dapat menggunakan beberapa model pergerakan *node*. Pada penelitian IoT/jaringan adhoc model mobilitas yang sangat populer dan umum digunakan adalah Random Way Point (RWP). RWP sangat cocok digunakan pada penelitian untuk melihat pengaruh kecepatan terhadap performa komunikasi antara *node* terutama yang diakibatkan oleh penurunan kualitas sinyal terima antar *node* yang disebabkan oleh pergerakan. Namun pada penelitian ini penelitian difokuskan terutama pada kondisi ekstrim dimana posisi *node* terhadap tetangganya selalu berubah. Dimana pada kondisi ini kemampuan perutean protokol benar-benar teruji untuk dapat menentukan rute terbaik yang akan digunakan untuk pengiriman data. Sehingga pada penelitian in untuk mencapai kondisi ekstrim tersebut maka digunakan model pergerakan Random Position (RP). Pada model ini maka untuk setiap waktu posisi *node* akan dibangkitkan secara random pada bidang *euclidean* 2 dimensi, seperti pada Gambar II.15.



Gambar II.15 Mobilitas Acak

$$Posisi\ Node(n) = rand(x_n), rand(y_n) \quad \text{(II.5)}$$

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad \text{(II.6)}$$

II.8.2 Model pembangkitan interest

Sementara pembangkitan *interest* sebuah konten berdasarkan (Breslau et al., 1999) dapat dimodelkan sebagai distribusi Zipf. Dalam sebuah jaringan NDN dimana terdapat aliran permintaan terhadap konten halaman web. Misalkan N adalah jumlah total halaman web di internet. Misalkan $P_N(i)$ merupakan probabilitas bersyarat, kedatangan permintaan halaman web, dimana permintaan halaman yang muncul ditujukan untuk halaman i. Misalkan bahwa setiap halaman di urutkan berdasarkan popularitas mereka dimana halaman ke i adalah urutan ke-i halaman paling populer. Nilai $P_N(i)$ didefinisikan untuk $i = 1,2,3, \dots, N,$ mendekati distribusi Zipf-like sebagai berikut

$$P_N(i) = \frac{\gamma}{i^\alpha} \quad \text{(II.5)}$$

dengan

$$\gamma = \left(\sum_{i=1}^{N} \frac{1}{i^\alpha}\right)^{-1} \quad \text{(II.6)}$$

Berdasarkan (Zipf, 1929) distribusi Zipf yang sesungguhnya memiliki nilai $\alpha = 1$ namun pada (Breslau et al., 1999) nilainya $0 < \alpha \leq 1$. Dimana setiap halaman di asumsikan independen terhadap halaman yang lain.

Sehingga model terhadap perutean *overhead* akan dibangun dari distribusi kedua pembangkitan di atas (pembangkitan keterhubungan, dan pembangkitan permintaan). Dimana perutean *overhead* memiliki kemungkinan muncul (i) pengecekan status dengan interval tetap, (ii) perubahan status link, dan (iii) pencarian rute reaktif dari pembangkitan pesan.

Bab III Model Efisiensi Perutean NDN

III.1 Model Sistem

Dalam mengukur seberapa baik algoritma perutean *Link State* (LS) dan *Geografical Routing* (GR) dilihat dari sudut pandang skalabilitas maka dibangunlah pemodelan sebagai berikut:

a. Pemodelan efisiensi penentuan parameter pendukung untuk perhitungan
b. Pemodelan *low-overhead* perutean

Dalam rangka meningkatkan skalabilitas perutean pada NDN dibangunlah algoritma perutean dan diseminasi informasi konten sebagai berikut:

a. Teknik *grid*
b. Minimalisir *signaling.*
c. Diseminasi informasi konten

Serta melakukan simulasi dari model efisiensi yang telah dibangun pada SK I

III.1.1 Pemodelan Efisiensi

Pada saat menjalankan protokol perutean dan mengirimkan pesan maka melibatkan paket kontrol, dan paket *overhead* sehingga perlu untuk dihitung seberapa besar efisiensi mekanisme protokol tersebut dalam mengirimkan pesan. Efisiensi pengiriman paket dapat di modelkan

$$\eta_o = \frac{R_o^{eff}}{R} = \frac{\frac{n_D - n_o}{t_o}}{R} \qquad \text{(III.1)}$$

n_D = Ukuran paket Data
n_o = Ukuran *overhead* dari paket data
R_o^{eff} = Rate transmisi efektif
R = Link Bandwidth (simetris)
t_o = Waktu mengirimkan paket

III.1.1.1. Model Perutean Geografis

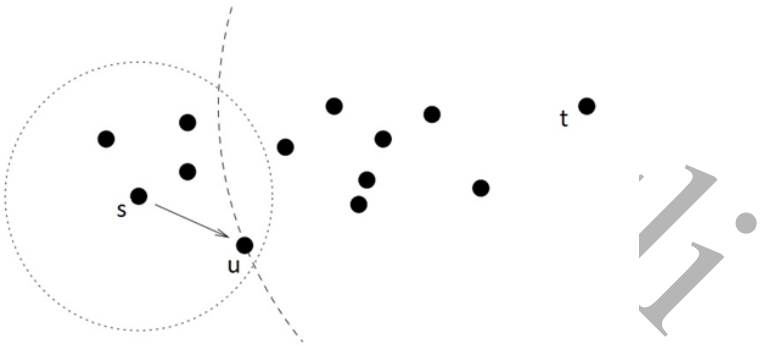
Dalam menjalankan mekanisme GR, kemampuan *node* dalam mendeteksi posisi atau biasa disebut *location-aware* adalah mutlak. Informasi terkait posisi dari

sebuah *node* bisa didapatkan dari GPS, infrared, dan trajektori dari sinyal radio masing-masing *node*.

Secara umum GR menggunakan algoritma *greedy* dalam menentukan jalur terbaik. Cara yang paling lazim dalam menentukan hop berikutnya adalah membandingkan jarak Euclidean setiap kemungkinan pilihan jalur menuju *node* tetangga sebagai simpul terdekat ke tujuan.

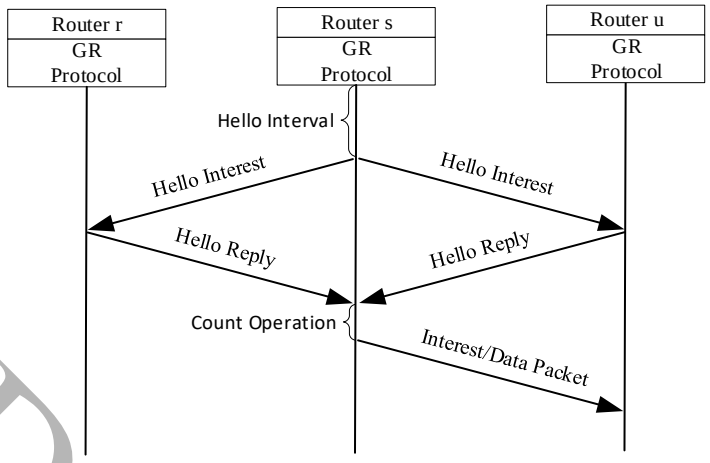
Definisi dari algoritma perutean *greedy* (Gao, 2009; Kleinberg, 2007) sebagaimana ditunjukkan pada Gambar II.1. dapat dijelaskan sebagai berikut. Pencarian *greedy* untuk hop selanjutnya dari sebuah graph tak berarah G untuk $f(s) \rightarrow f(t)$ pada metric space (X, d) merupakan mapping $f: V(G) \rightarrow X$ dengan syarat sebagai berikut: untuk setiap pasangan vertex s, $t \in V(G)$ terdapat vertex u bertetanggaan dengan s sehingga $d(f(u),f(t)) = \perp$ untuk setiap pasangan *node* menuju t, bertetanggaan dengan s.

Pemilihan hop selanjutnya dilakukan pada setiap *node* untuk memilih biaya minimum lokal yang diharapkan menjadi solusi biaya minimum secara keseluruhan.

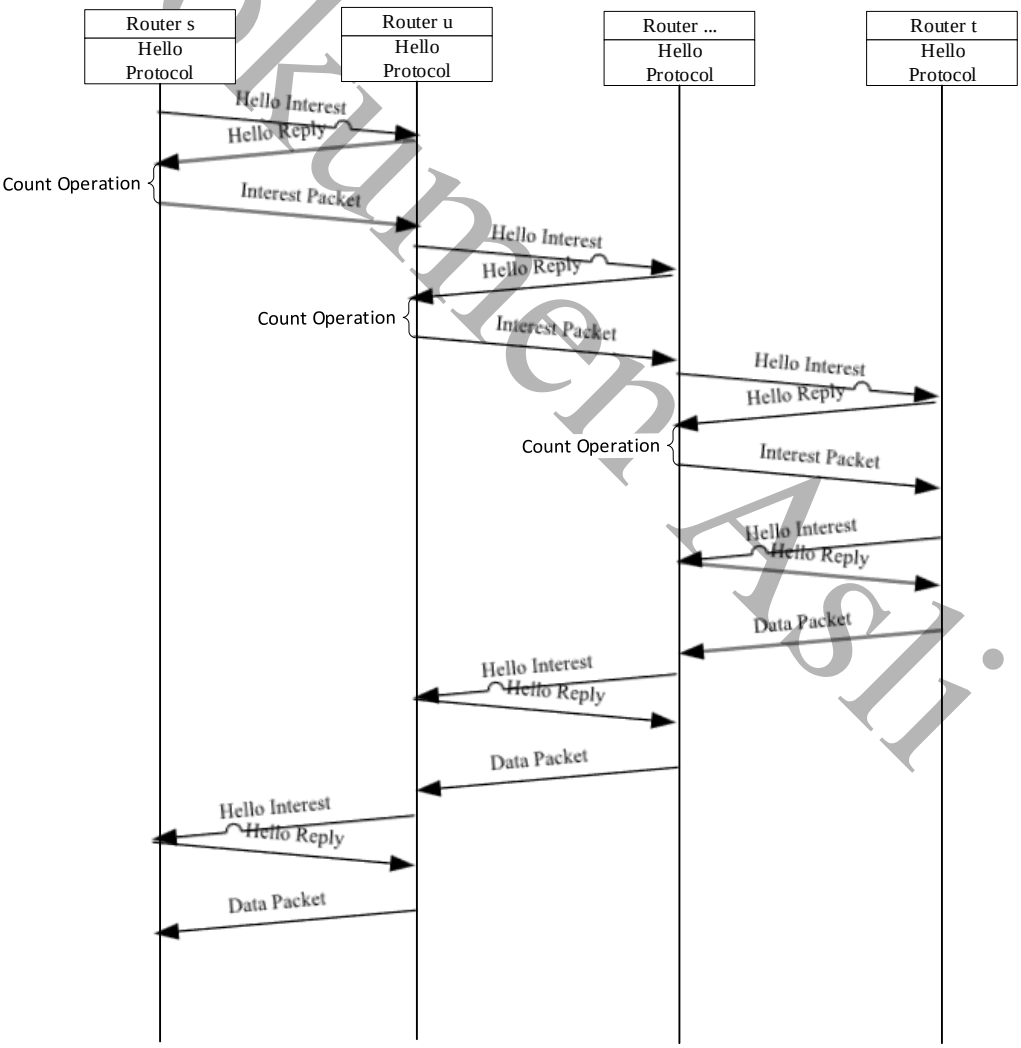


Gambar III.1 Pencarian rute secara *Greedy* (Gao, 2009)

Jika digambarkan dalam bentuk time diagram proses.

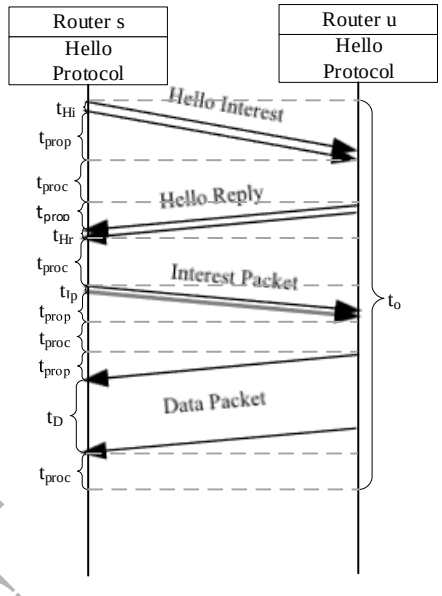


Gambar III.2 Mekanisme Hello Protokol Pada GR



Gambar III.3 Mekanisme Pengiriman Paket

A. Perutean Geografis (1 Hop, eror free)



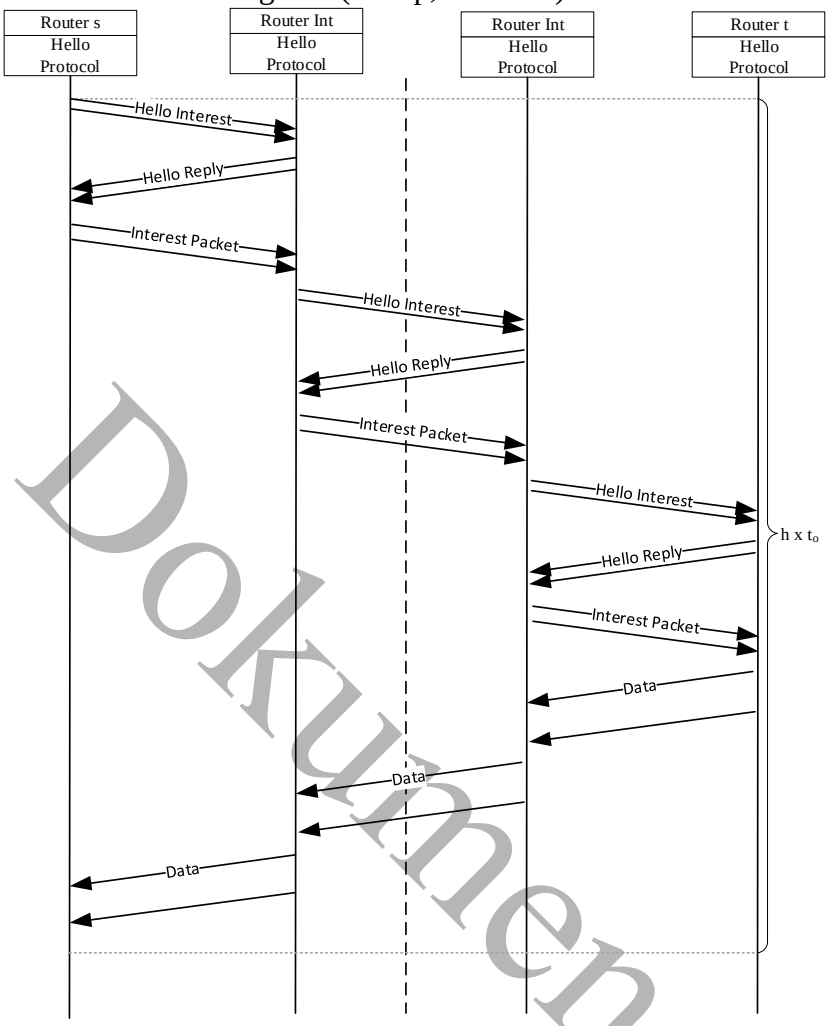
Gambar III.4 Mekanisme Perutean Geografis (1 Hop, eror free)

$$t_o = 4tprop + 4tproc + t_{Hi} + t_{Hr} + t_{Ip} + t_D \quad \text{(III.2)}$$
$$= 4tprop + 4tproc + \frac{n_{Hi}}{R} + \frac{n_{Hr}}{R} + \frac{n_{Ip}}{R} + \frac{n_D}{R}$$

$$R_o^{eff} = \frac{Bits\ of\ Information}{time\ to\ deliver} = \frac{n_D - n_o}{t_o} \quad \text{(III.3)}$$

$$\eta_o = \frac{R_o^{eff}}{R} = \frac{\frac{n_D - n_o}{t_o}}{R} \quad \text{(III.4)}$$
$$= \frac{1}{R} \times \frac{n_D - n_o}{4tprop + 4tproc + \frac{n_{Hi}}{R} + \frac{n_{Hr}}{R} + \frac{n_{Ip}}{R} + \frac{n_D}{R}}$$
$$= \frac{n_D - n_o}{4R(tprop + tproc) + n_{Hi} + n_{Hr} + n_{Ip} + n_D} \times \frac{\frac{1}{n_D}}{\frac{1}{n_D}}$$
$$= \frac{1 - \frac{n_o}{n_D}}{1 + \frac{n_{Hi}}{n_D} + \frac{n_{Hr}}{n_D} + \frac{n_{Ip}}{n_D} + \frac{4R(tprop + tproc)}{n_D}}$$

B. Perutean Geografis (h Hop, eror free)



Gambar III.5 Mekanisme Perutean Geografis (h Hop, eror free)

$$t\,hHop = h \times t_o \qquad \text{(III.5)}$$
$$= h\left(4tprop + 4tproc + t_{Hi} + t_{Hr} + t_{Ip} + t_D\right)$$
$$= h\left(4tprop + 4tproc + \frac{n_{Hi}}{R} + \frac{n_{Hr}}{R} + \frac{n_{Ip}}{R} + \frac{n_D}{R}\right)$$

$$R_o^{eff} = \frac{Bits\ of\ Information}{time\ to\ deliver} = \frac{n_D - n_o}{h.t_o} \qquad \text{(III.6)}$$

$$\eta_o = \frac{R_o^{eff}}{R} = \frac{\frac{n_D - n_o}{h.t_o}}{R} \qquad \text{(III.7)}$$
$$= \frac{1}{R} \times \frac{n_D - n_o}{h\left(4tprop + 4tproc + \frac{n_{Hi}}{R} + \frac{n_{Hr}}{R} + \frac{n_{Ip}}{R} + \frac{n_D}{R}\right)}$$

$$= \frac{1}{h} \times \frac{n_D - n_o}{4R(tprop + tproc) + n_{Hi} + n_{Hr} + n_{Ip} + n_D} \times \frac{\frac{1}{n_D}}{\frac{1}{n_D}}$$

$$= \frac{1}{h} \times \frac{1 - \frac{n_o}{n_D}}{1 + \frac{n_{Hi}}{n_D} + \frac{n_{Hr}}{n_D} + \frac{n_{Ip}}{n_D} + \frac{4R(tprop + tproc)}{n_D}}$$

$$= \frac{1}{h} \times \eta_o(1 hop GR)$$

C. Perutean Geografis (h Hop, dengan eror)

Jika n_t adalah jumlah transmisi paket yang dibutuhkan untuk mengirimkan paket hingga sukses; maka $n_t = i$ transmisi dibutuhkan jika $i - 1$ eror dan transmisi ke i berhasil. Sehingga

$P[n_t = i] = (1 - P_p)P_p^{i-1}$ untuk $i = 1, 2, 3, \ldots$

Dimana

$$E[n_t = 1] = \sum_{i=1}^{\infty} i(1 - P_p)P_p^{i-1} \quad \text{(III.8)}$$

$$\mu_p = (1 - P_p) \sum_{i=1}^{\infty} i.P_p^{i-1} \quad \text{(III.9)}$$

$$\mu_p = (1 - P_p)[1 + 2P_p + 3P_p^2 + 4P_p^3 + \cdots]$$

$$P_p.\mu_p = (1 - P_p)[P_p + 2P_p^2 + 3P_p^3 + 4P_p^4 \ldots]$$

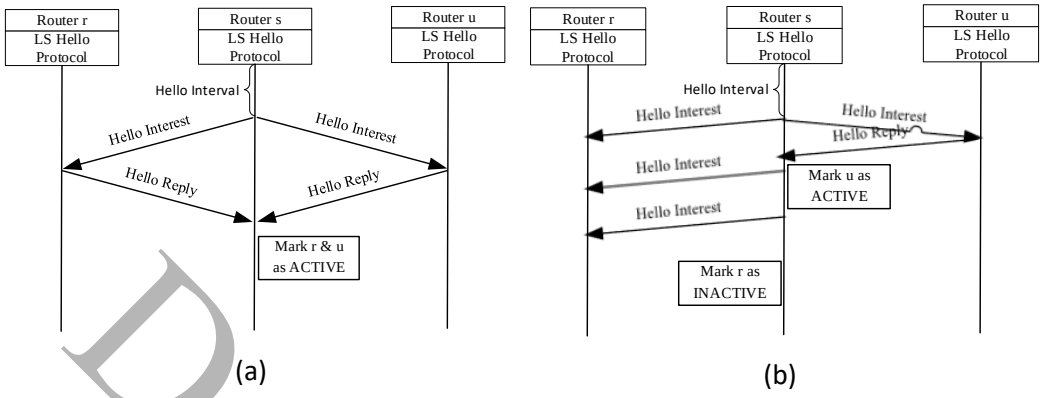
$$\mu_p(1 - P_p) = (1 - P_p)[1 + P_p + P_p^2 + P_p^3 + \cdots]$$

$$\mu_p = \frac{1}{1 - P_p} \quad \text{(III.10)}$$

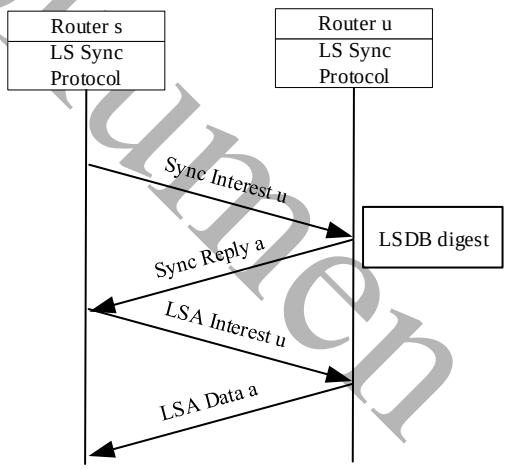
III.1.1.2. Model Perutean *Link State*

Dalam rangka menjalankan pemilihan rute terhadap jalur terbaik, pada perutean LS, setiap *node* harus memetakan seluruh kemungkinan jalur pada jaringan. Proses yang pertama kali harus dilakukan adalah memeriksa keberadaan *node* tetanga. Proses tersebut akan terus berlangsung secara periodik selama sebuah *node* aktif. Secara umum, proses tersebut dilakukan dengan cara mengirimkan sebuah paket bernama *hello messages* secara broadcast ke seluruh interface yang terhubung pada *node*, sebagaimana dapat dilihat pada Gambar III.6. Ketika *Router* s mengirimkan paket *"Hello"* dan mendapatkan balasan *"Hello Reply"* dari *Router* r dan u, maka

Router s akan mencatat pada route table bahwa *Router* r dan u active. Jika tidak mendapatkan balasan *Router* s akan mengirimkan Hello beberapa kali yang pada akhirnya memberikan tanda Inactive jika belum juga mendapatkan balasan

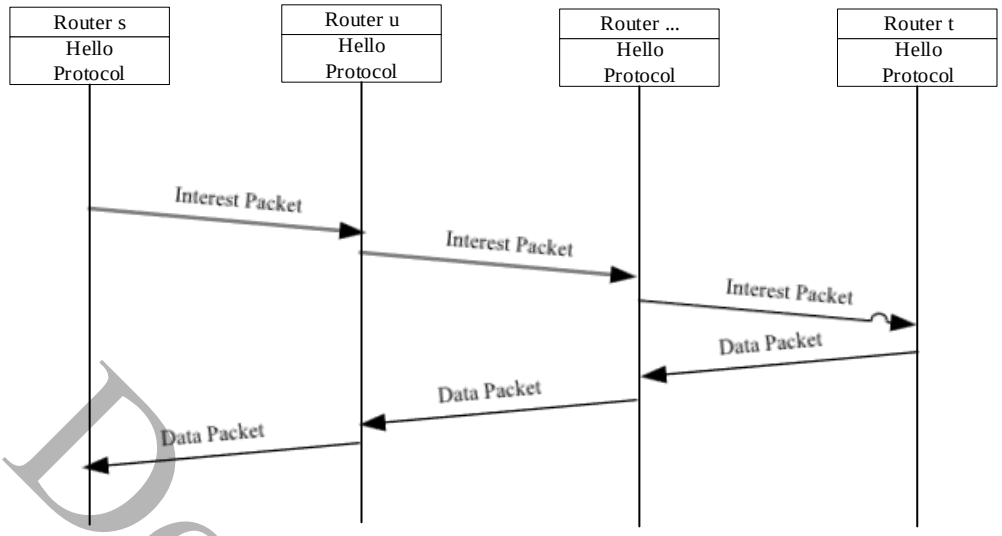


Gambar III.6 Mekanisme *Hello* Pada LS (a) *Router* Aktif; (b) *Router* Inactive



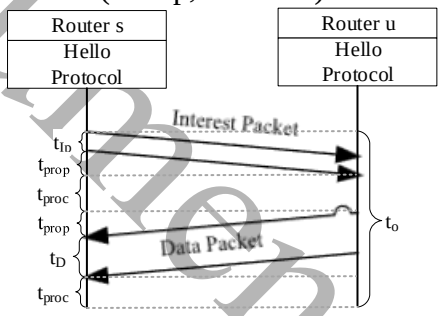
Gambar III.7 Mekanisme Desiminasi LSA pada LS

Setelah inisialisasi pada *node* tetangga selesai *node* NDN akan melakukan pertukaran data database (LSA) yang berisis peta jaringan dari sudut pandang *node* tetangganya, sebagaimana dapat dilihat pada Gambar III.7. *Node* s akan mengirimkan permintaan sinkronisasi *sync interest* yang kemudian dibalas oleh *node* u dengan pesan sync reply berisi digest dari tabel perutean *node* u. *Node* s akan mencocokkan dengan database dirinya jika terdapat perbedaan maka akan dilanjutkan dengan mengitimkan LSA Interest, kemudian *node* u akana membalas dnegan mengirimkan LSA data kepada *node* s.



Gambar III.8 Mekanisme Pengiriman Informasi pada LS

A. Perutean *Link State* (1 Hop, eror free)



Gambar III.9 Mekanisme perutean Link State (1 Hop, tanpa eror)

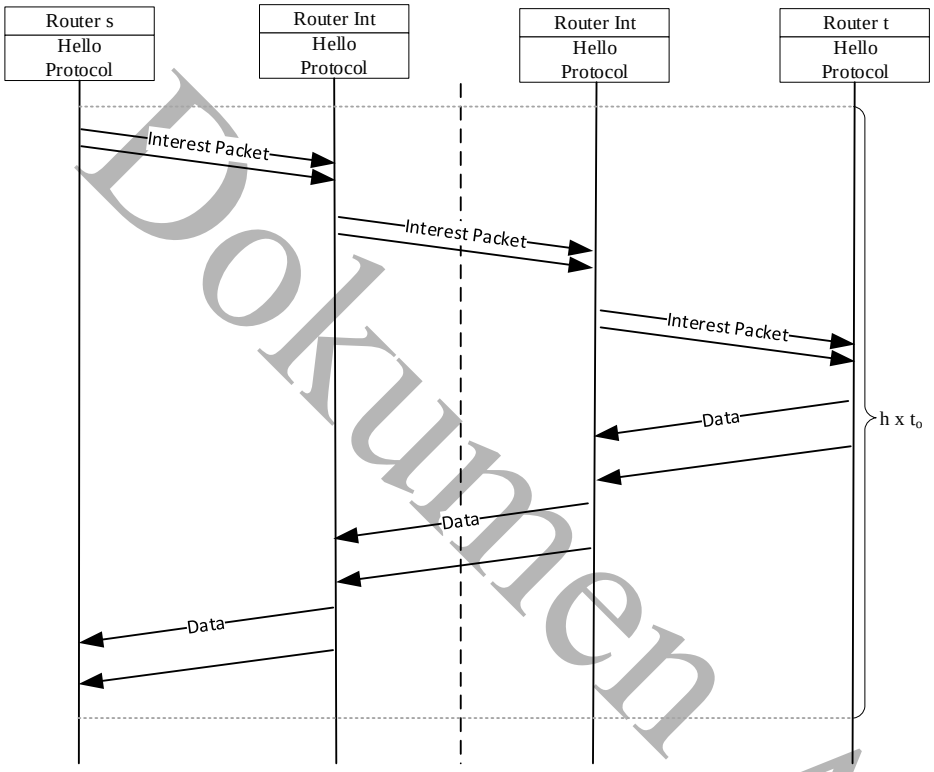
$$t_o = 2tprop + 2tproc + t_{Ip} + t_D$$
$$= 4tprop + 4tproc + \frac{n_{Ip}}{R} + \frac{n_D}{R} \quad \text{(III.11)}$$

$$R_o^{eff} = \frac{Bits\ of\ Information}{time\ to\ deliver} = \frac{n_D - n_o}{t_o}$$

$$\eta_o = \frac{R_o^{eff}}{R} = \frac{\frac{n_D - n_o}{t_o}}{R}$$
$$= \frac{1}{R} \times \frac{n_D - n_o}{2tprop + 2tproc + \frac{n_{Ip}}{R} + \frac{n_D}{R}}$$
$$= \frac{n_D - n_o}{2R(tprop + tproc) + n_{Ip} + n_D} \times \frac{\frac{1}{n_D}}{\frac{1}{n_D}} \quad \text{(III.12)}$$

$$= \frac{1 - \frac{n_o}{n_D}}{1 + \frac{n_{Ip}}{n_D} + \frac{2R(tprop + tproc)}{n_D}}$$

B. Perutean *Link State* (h Hop, eror free)



Gambar III.10 Mekanisme perutean Link State (h Hop, tanpa eror)

$$t_o = 2tprop + 2tproc + t_{Ip} + t_D$$
$$= 4tprop + 4tproc + \frac{n_{Ip}}{R} + \frac{n_D}{R} \quad \text{(III.13)}$$
$$R_o^{eff} = \frac{Bits\ of\ Information}{time\ to\ deliver} = \frac{n_D - n_o}{h.t_o}$$
$$\eta_o = \frac{R_o^{eff}}{R} = \frac{\frac{n_D - n_o}{h.t_o}}{R}$$
$$= \frac{1}{R} \times \frac{n_D - n_o}{h\left(2tprop + 2tproc + \frac{n_{Ip}}{R} + \frac{n_D}{R}\right)}$$
$$= \frac{1}{h} \times \frac{n_D - n_o}{2R(tprop + tproc) + n_{Ip} + n_D} \times \frac{\frac{1}{n_D}}{\frac{1}{n_D}}$$

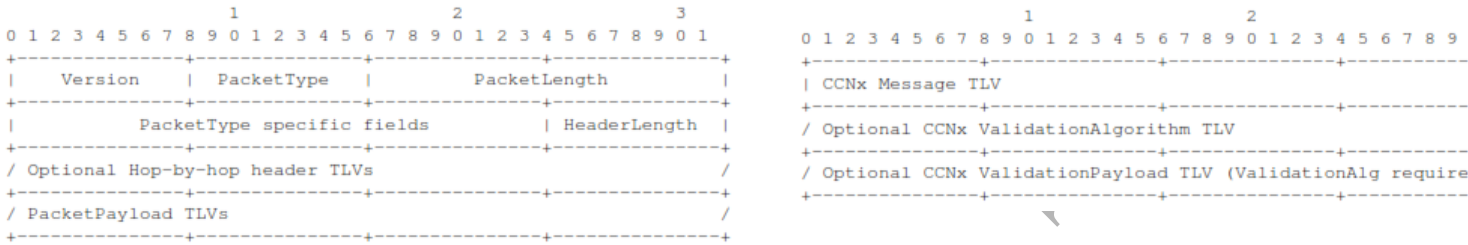
$$= \frac{1}{h} \times \frac{1 - \frac{n_o}{n_D}}{1 + \frac{n_{Ip}}{n_D} + \frac{2R(tprop + tproc)}{n_D}}$$
$$= \frac{1}{h} \times \eta_o(1HopLS) \quad \text{(III.14)}$$

C. Perutean *Link State* (h Hop, dengan eror)

$$E\eta'_o = \frac{R_o^{eff}}{R} = \frac{\frac{n_D - n_o}{h.t'_o}}{R}$$
$$= \frac{1}{R}(1 - P_d) \times \frac{n_D - n_o}{h\left(2tprop + 2tproc + \frac{n_{Ip}}{R} + \frac{n_D}{R}\right)}$$
$$= \frac{1}{h}(1 - P_d) \times \frac{n_D - n_o}{2R(tprop + tproc) + n_{Ip} + n_D} \times \frac{\frac{1}{n_D}}{\frac{1}{n_D}}$$
$$= \frac{1}{h}(1 - P_d) \times \frac{1 - \frac{n_o}{n_D}}{1 + \frac{n_{Ip}}{n_D} + \frac{2R(tprop + tproc)}{n_D}}$$
$$= \frac{1}{h}(1 - P_d) \times \eta_o(1HopLS) \quad \text{(III.15)}$$

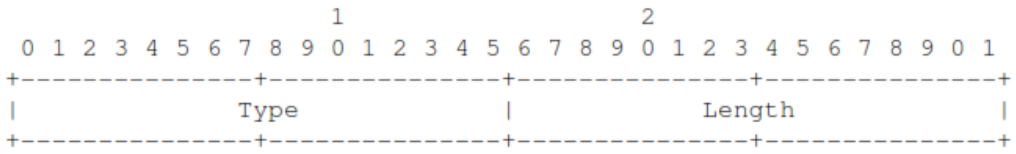
III.1.1.3. Format Paket Umum

Secara umum NDN paket format di deskripsikan seperti pada Gambar II.11. Hal tersebut sesuai dengan draft standard dari ICN (Mosko et al., 2019).



Gambar III.11 Overall Packet format

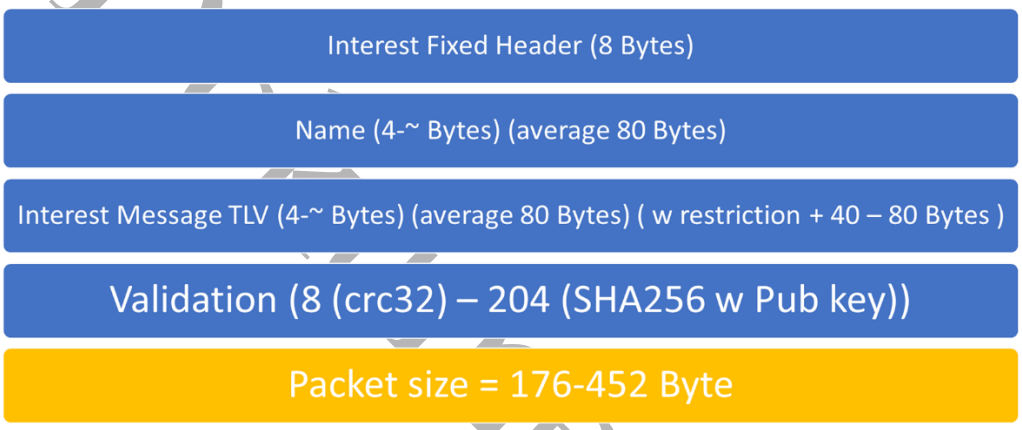
Dengan tujuan untuk efisiensi disebabkan variasi panjang paket, maka CCNX message menggunakan mekanisme Type Lengt Value (TLV). Dengan mekanisme TLV variasi panjang paket dan tipe dapat diakomodir sebagaimana dapat dilihat pada Gambar II.12.



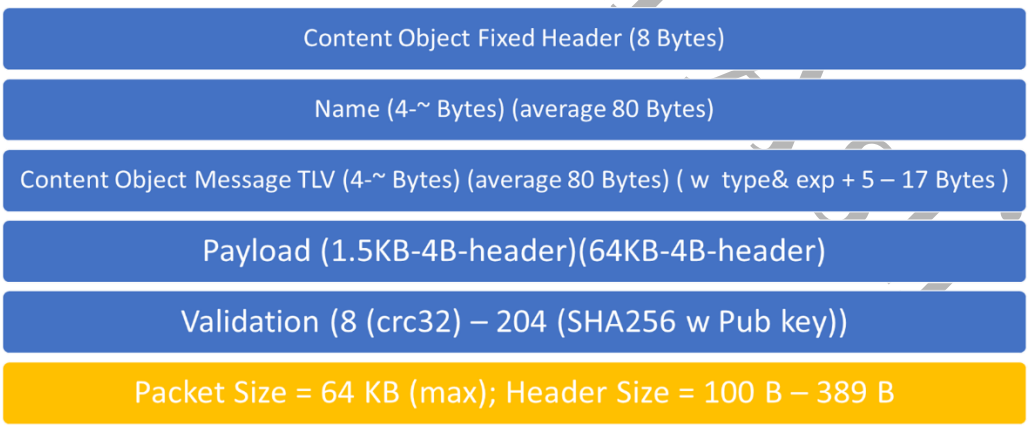
Gambar III.12 TLV Format

Berdasarkan standar tersebut diatas maka dihitunglah kebutuhan panjang paket interest, data dan hello message, sebagai berikut:

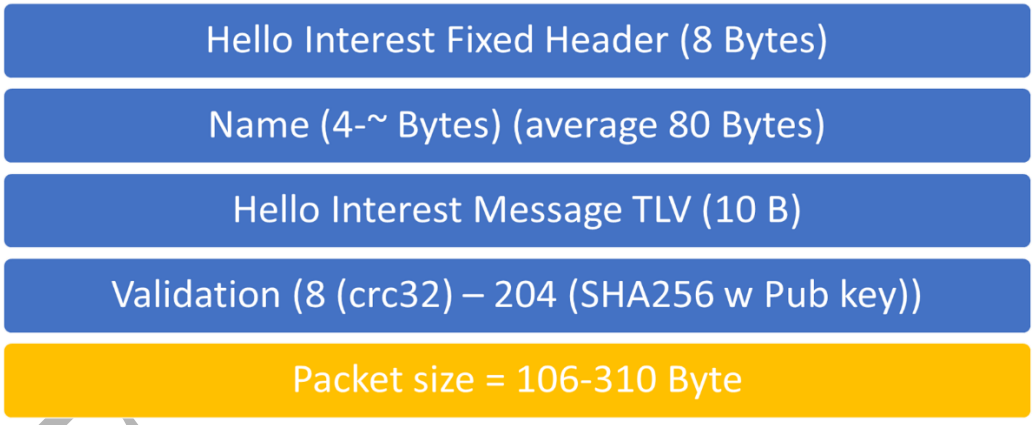
III.1.1.4. Interest Message



III.1.1.5. Data Message

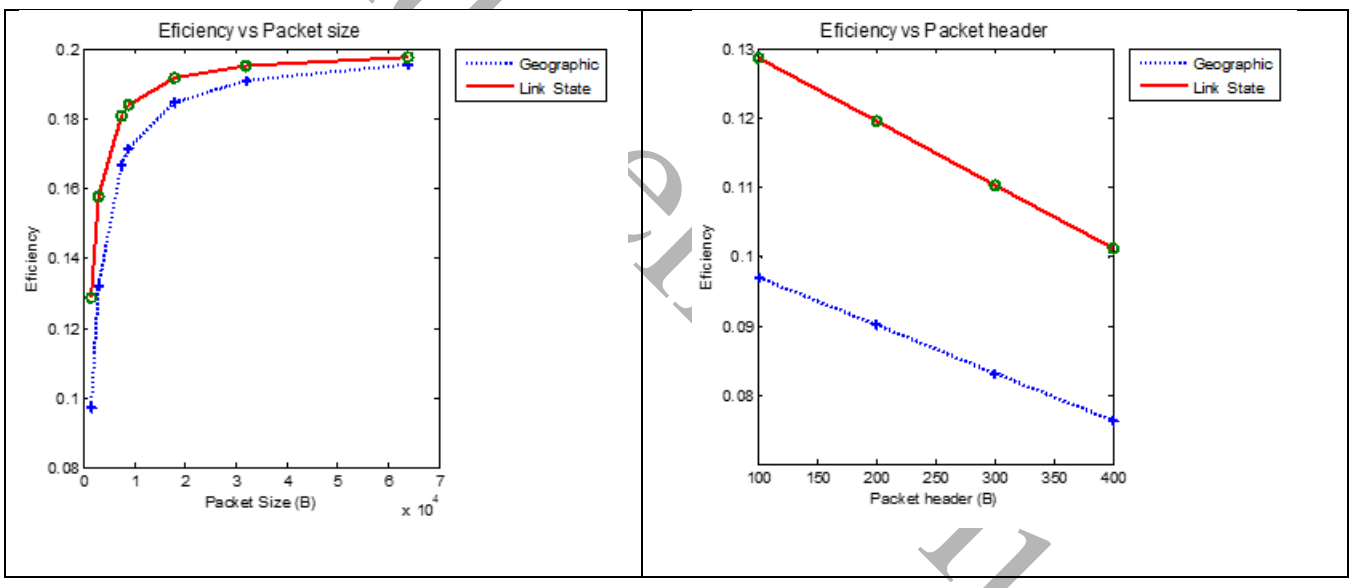


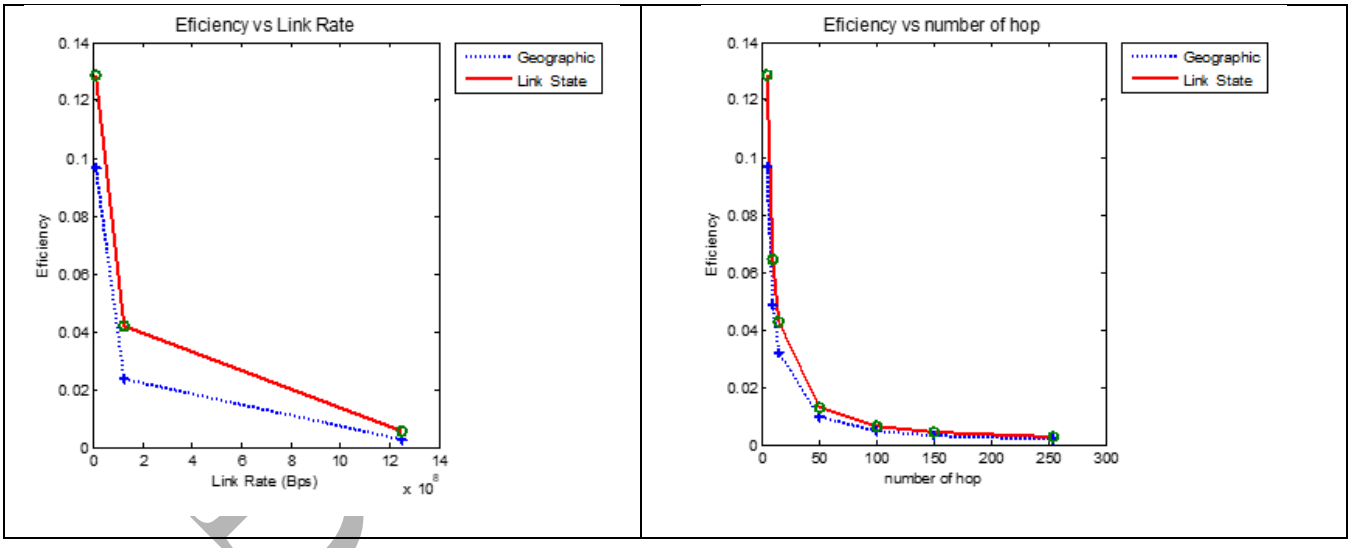
III.1.1.6. Hello Message



III.1.2 Hasil Simulasi

Dari model yang di bangun untuk melihat seberapa efisienkah protokol perutean GR dan LS dalam menangani perubahan yang terjadi di jaringan maka dilakukan simulasi sebagaimana dapat dilihat pada Gambar III.13





Gambar III.13 Hasil Simulasi Efisiensi

Bab IV
Protokol dan Arsitektur Perutean untuk Skalabilitas dan Mobilitas

IV.1 *Overhead* Perutean

Mekanisme perutean pada jaringan *wireless* merupakan salah satu tema riset yang cukup intens untuk di teliti terutama pada lingkungan ad-hoc, manet, vanet dll. Mayoritas dari algoritma perutean yang diusulkan seperti DSR (Johnson et al., 2007), AODV (Perkins and Royer, 1999), sudah berhasil dalam menyediakan perutean yang reliable pada jaringan ad-hoc.

Fokus pada reliabilitas dan skalabilitas dari perutean protokol maka akan dibangun sebuah protokol perutean dengan performa sbb:

a. Memiliki pengetahuan terhadap lokasi *node*. Perutean berbasis geografis memiliki keunggulan pada reliabilitas, dan dukungan kebutuhan terhadap aplikasi yang membutuhkan informasi terhadap lokasi
b. Mekanisme perutean dipelihara secara terdistribusi. Tidak ada *node* yang akan menjadi bottleneck dalam proses perutean. Ketergantungan pada *server* lokasi terpusat akan meningkatkan beban *node* tersebut, single point of failure, serta meningkatkan kongesti di sekitar *server*.
c. Kegagalan/perubahan status pada sebuah *node* tidak mempengaruhi *node* lain. Pada jaringan ad-hoc kegagalan/perubahan status *node* akan lebih kerap terjadi
d. Komunikasi terkait lokasi dengan *node* tetangga cukup dilakukan secara lokal. Ketika sebuah source dan destinasi *node* saling berdekatan, maka proses pencarian jalur menjadi operasi yang harus lebih sederhana.
e. Komunikasi signaling untuk setiap pencarian rute harus meminimalisir *overhead* dan *flooding* pesan. Untuk mempertahankan pilihan rute diperlukan signaling antar *node*. Dalam hal skalabilitas yang baik maka proses tersebut harus se efisien mungkin.
f. Update status dilakukan mengikuti perilaku *node*. Kedinamisan *node* pada jaringan ad hoc dapat memiliki variasi yang besar maka update harus dilakukan secara adaptif

Dalam rangka mencapai desain goal tersebut dibutuhkan desain yang skalable untuk jaringan NDN yang memiliki karakteristik sebagai berikut:

a. Tujuan dari perutean NDN adalah mencari keberadaan konten.
b. Mayoritas komunikasi yang terjadi klien – klien atau komunikasi *peer to peer*
c. Untuk meningkatkan availibility juga memungkinkan untuk dibangun Multipath route to client
d. Perutean berbasis Nama
e. Perutean akan menggunakan mekanisme hibrid (reaktif dan proaktif)
f. Perutean akan menggunakan informasi geografis
g. *Node* memiliki sensor yang dibutuhkan dalam algoritma perutean (GPS, Accelerometer, gyrometer, pengukur daya baterei)
h. Mekanisme perutean akan disederhanakan dengan beberapa konsensus (luas area, jumlah hop, cakupan kluster, dll)

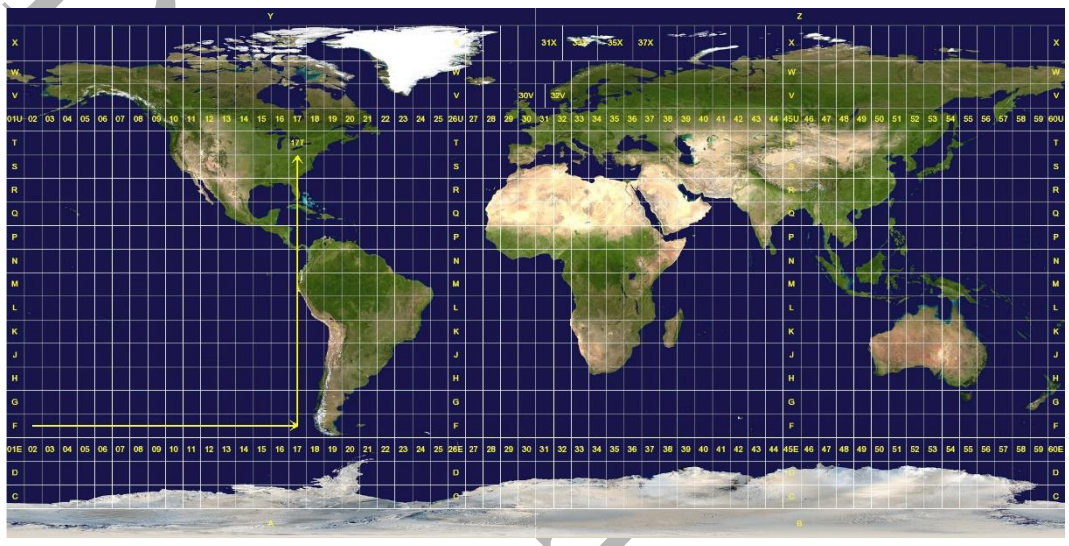
IV.2 Perutean GSNR

Dengan tujuan meningkatkan skalabilitas pada jaringan NDN maka diperlukan algoritma perutean yang sederhana namun dapat memberikan performa yang baik dalam melakukan komunikasi. Protokol perutean yang diusulkan ini diberi nama *Grid-based Scalable NDN Routing* (GSNR). Mekanisme GSNR menggunakan perutean berbasis grid. Sebuah *node* bertindak sebagai *Cluster Head* (CH) dan *Cluster Gateway* (CG) di setiap area grid.

Teknik grid perutean membagi area topografi jaringan menjadi grid dengan area yang telah ditentukan sebelumnya. Cakupan dan identitas area grid telah ditentukan sebelumnya dan disepakati sebagai konsensus untuk menyederhanakan mekanisme perutean. Konsensus digunakan untuk meminimalkan *overhead* untuk meningkatkan skala jaringan. Kami membagi area menjadi persegi panjang seragam sesuai dengan *Universal Transverse Mercator* (UTM) (Buchroithner and Pfahlbusch, 2017). Grid id diberikan secara unik, sehingga protokol perutean dapat menggunakan aturan ini untuk area di seluruh dunia.

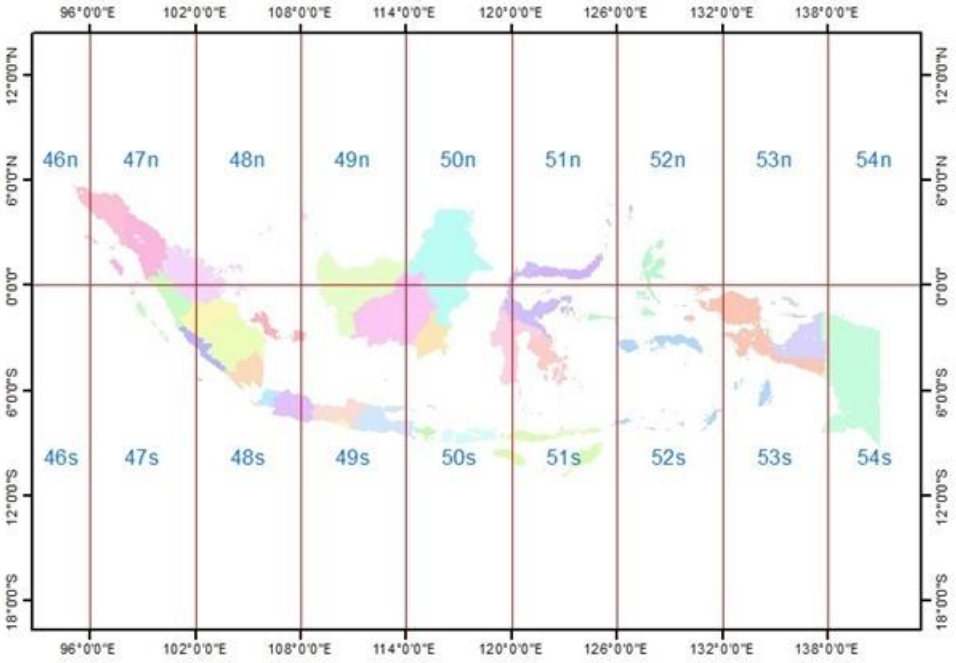
A. Koordinat sistem

Sistem Proyeksi Koordinat UTM (*Universal Transverse Mercator*) adalah rangkaian proyeksi Transverse Mercator yang berlaku global. Pada sistem proyeksi ini bumi dibagi kedalam 60 bagian zona. Setiap zona mencangkup 6 derajat bujur (longitude) dan memiliki meridian tengah tersendiri, dapat dilihat pada Gambar IV.1. Berbeda dengan koordinat geografi yang satuan unitnya adalah derajat, koordinat UTM menggunakan satuan unit meter. Setiap zona memiliki panjang x sebesar 500.000 meter dan panjang nya sebesar 10.000.000 meter.



Gambar IV.1 Koordinat Global UTM

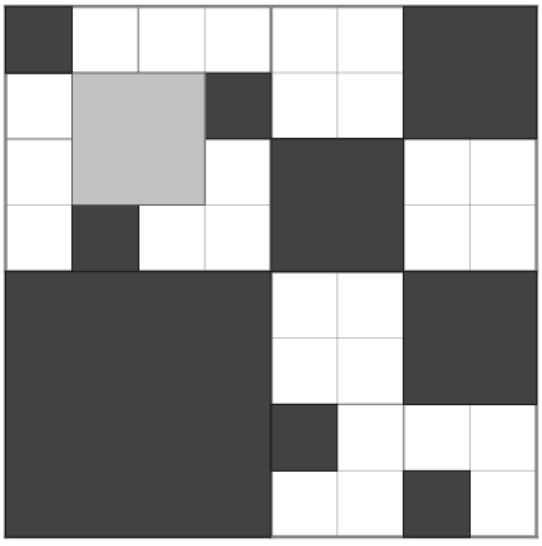
Proyeksi ini menjadi dasar koordinat sistem global yang pada awalnya dikembangkan untuk keperluan militer, namun sekarang sudah dipakai lebih luas. Sehingga, zona 1 pada koordinat UTM dimulai dari 1800 BB-1740BB, kemudian dilanjutkan dengan zona 2 yang dimulai dari 1740 BB-1680 BB, zona 3 dimulai dari 1680 BB-1620 BB, dst… sedangkan untuk batas lintang dibagi berdasarkan nilai 8 derajat. Untuk Indonesia yang berada pada posisi 900BT - 1440BT dan 110LS - 60LU terbagi ke dalam 9 zona UTM yaitu zona 46 – 54, dapat dilihat pada Gambar IV.2.



Gambar IV.2 UTM Area Indonesia

B. Grid perutean

Teknik grid perutean digunakan dengan membagi bidang-bidang topografi jaringan menjadi semacam grid yang luasnya sudah ditentukan, dapat dilihat pada Gambar IV.3. Hal tersebut dilakukan dalam rangka untuk menyepakati luasan area yang akan digunakan sebagai identitas dalam membantu mekanisme perutean, menyederhanakan proses, meminimalisir *overhead* dengan tujuan meningkatkan skalabilitas.



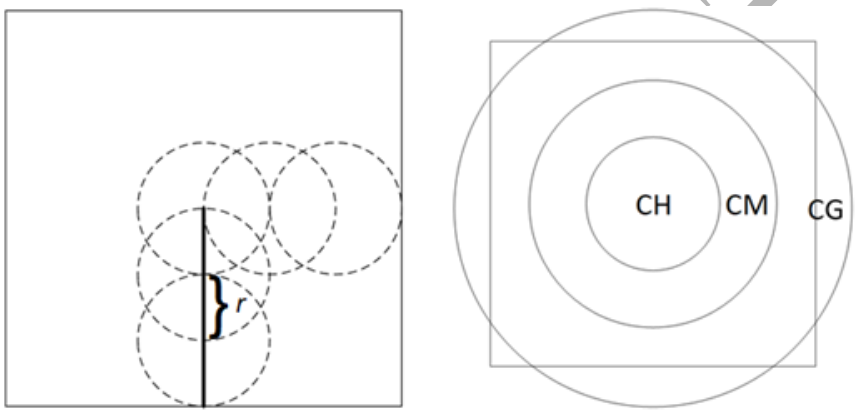
Gambar IV.3 Grid area

Grid yang akan dibangun dalam mewujudkan perutean yang skalable akan menggunakan mekanisme multi-hop grid dapat dilihat pada Gambar.IV.4. Pada skema multi hop grid maka setiap *node* akan memiliki minimal satu dari salah satu status:

- Head Grid (HG)
- Grid Gateway (GG)
- Grid Member (GM)

HG memiliki fungsi sebagai koordinator *node* pada grid. Setiap *node* yang melakukan pencarian rute terhadap konten akan mengirimkan permintaan/interest palet kepada GM. GM memiliki database konten lokal grid. GG berfungsi sebagai jembatan komunikasi inter-grid (dengan grid tetangga). GG akan meneruskan komunikasi ke *node* tetangga jika dibutuhkan. GM adalah *node-node* yang tidak memiliki status HG ataupun GG.

Sehingga dengan penentuan jumlah K-hop komunikasi grid didapat jumlah grid untuk setiap UTM area. Nilai K akan sebanding dengan delay yang muncul akibat komunikasi reaktif. Namun nilai K berbanding terbalik terhadap nilai *overhead* yang muncul pada perutean reaktif untuk komunikasi inter-grid. Sebagaimana dalam contoh kasus yang ditunjukkan pada Gambar IV.4 menggunakan nilai K = 2 hop yang artinya komunikasi di dalam grid (*intra grid*) dari *grid member* ke *head grid* dan atau *head grid* ke *grid gateway* maksimum melalui 2 hop. Jika r adalah radius transmisi dari sebuah *node*, sehingga didapatkan untuk K=2 luasan area grid terhadap nilai r nya adalah 6r x 6r . Maka didapatkan hubungan grid area persamaan IV.1 dan jumlah grid yang terbentuk persamaan IV.2.



Gambar IV.4 Multi-hop Grid

$$Grid\ area = \left((K+1)2r\right)^2 \quad \text{(IV.1)}$$

$$\sum Grid = \frac{UTM\ area}{Grid\ area} = \frac{5 \times 10^{12}}{\left((K+1)2r\right)^2} \quad \text{(IV.2)}$$

HG dan GG sangat intens terlibat dalam komunikasi. Hampir setiap komunikasi

C. Penentuan status *node*

Dalam menentukan status dari *node* pada suatu maka dibutuhkan sebuah bobot sentralitas. sehingga kita dapat melakukan pemeringkatan. Pada struktur grid pada bidang 2 dimensi maka masing-masing grid akan memiliki hubungan dengan grid yang lain sebagaimana dapat dilihat pada Gambar IV.5. Setiap Current Grid akan memiliki paling tidak 8 Neighbor Grid. Sehingga untuk setiap Current Grid perlu menentukan *gateway* untuk setiap Neighbor Grid. Maka setiap Current Grid akan memiliki alokasi 1 HG dan 8 GG yang bisa saja satu *node* dalam current grid memiiki lebih dari satu status tersebut

Neighbor Grid 1	Neighbor Grid 2	Neighbor Grid 3
Neighbor Grid 4	Current Grid	Neighbor Grid 5
Neighbor Grid 6	Neighbor Grid 7	Neighbor Grid 8

Gambar IV.5 Struktur Grid

Dalam menentukan HG dan GG untuk mengoptimalkan mekanisme perutean di jaringan maka diperlukan perhitungan sentralitas *node* dari sudut pandang *cluster*isasi area. Dalam menjalankan komunikasi jaringan ad-hoc dibutuhkan HG dengan preferensi berada di titik sentral dari grid dan GG yang merupakan *node* terdekat dengan grid tetangga. Untuk setiap grid (sudut pandang area dua dimensi) seperti pada Gambar IV.6, paling tidak diperlukan 1 sentralitas HG (dinotasikan

sebagai C_{Gv}) dan 8 sentralitas GG (dinotasikan sebagai I_{nGv}). Dalam menentukan nilai sentralitas dapat digunakan persamaan sebagai berikut:

$$C_{Gv} = \sqrt{(X_T - X_c)^2 + (Y_T - Y_c)^2}$$
$$I_{nGv} = \sqrt{(X_T - X_{nc})^2 + (Y_T - Y_{nc})^2}$$

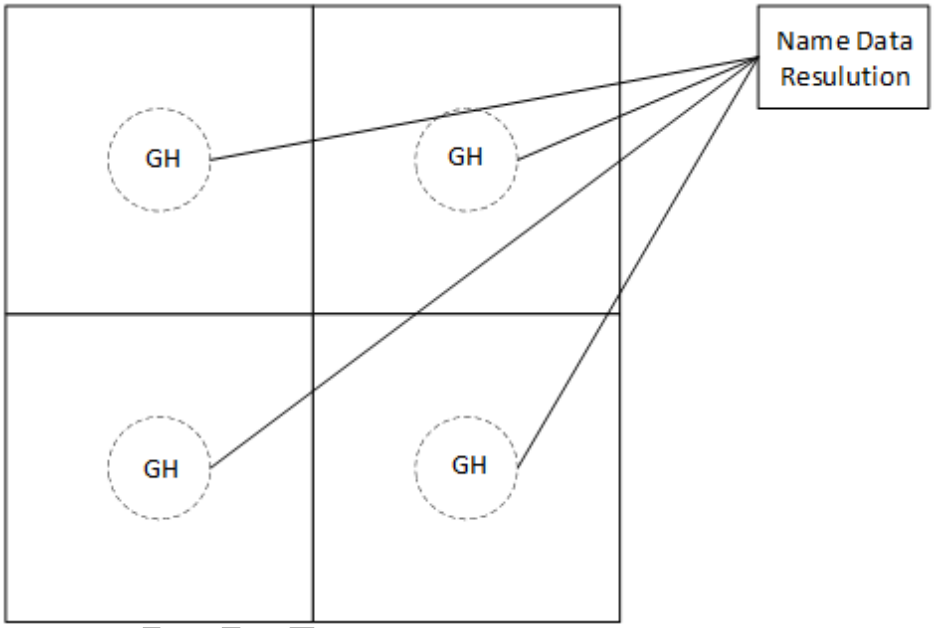
Dimana untuk setiap nilai C_{Gv} dan I_{nGv} untuk *cluster* tertentu akan di lakukan pemeringkatan. *Node* dengan nilai C_{Gv} terkecil akan menjadi HG sedangkan *node* dengan I_{nGv} terkecil akan menjadi *grid gateway* untuk grid ke-n

IV.2.1 Protokol Perutean

Dengan tujuan meningkatkan skalabilitas pada jaringan NDN maka diperlukan protokol perutean yang sederhana namun dapat memberikan performa yang baik dalam melakukan komunikasi. Selanjutnya akan dijabarkan bagaimana mekanisme perutean berjalan dikaitkan dengan name resolution, intra-grid dan inter-grid perutean

Pada proses *request* dan pengiriman data, di jaringan NDN, diperlukan pengetahuan tentang informasi keberadaan konten dan lokasinya. Mekanisme diseminasi keberadaan konten biasanya ditangani oleh sebuah sistem yang biasa disebut Name Resolution System (NRS). Pada saat sebuah *node* ingin mengirimkan request terkait konten tertentu, dan tidak memiliki informasi tentang keberadaan konten tersebut, maka *node* dapat menghubungi NRS untuk meminta informasi keberadaan konten.

Pada protokol GSNR terdapat dua mekanisme dalam mendapatkan informasi keberadaan konten. Untuk informasi lokal/intra grid maka informasi keberadaan konten yang ada di dalam cakupan grid bisa didapatkan pada HG. Untuk informasi global/inter grid maka informasi konten di sediakan oleh NRS. NRS yang digunakan oleh CGR dapat yang bersifat distribusi (Distributed Hash Table (DHT)/ Bloom Filter), atau yang bersifat sentral (NDN Domain Name *Server*) sebagaimana dapat dilihat pada gambar IV.6.



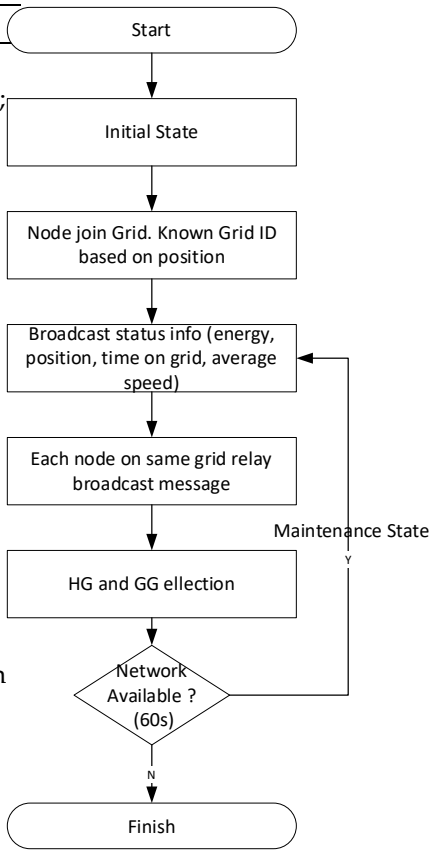
Gambar IV.6 NDN *Name Resolution Intra-Grid Routing*

Pada fase awal, setiap *node* membroadcast pesan info tentang posisi dan ID gridnya. Dari info tersebut, setiap *node* menghitung sentralitas HG dan GG untuk memilih HG dan GG untuk setiap grid tetangga. Ketika HG dan GG untuk setiap grid dipilih, maka keadaan awal selesai. Selama komunikasi pada jaringan terjadi, *node* HG dan GG harus diperbarui secara berkala dimana masuk dalam fase pemeliharaan. Pada fase pemeliharaan, secara periodik (60 detik), setiap *node* memperbaharui info posisinya. Kemudian dengan data pembaharuan posisi, setiap *node* mengupdate status HG dan GG. Fase awal dan pemeliharaan dapat dijabarkan pada Algoritma 1

Algorithm 1: Initial and Maintenance State

Initial State

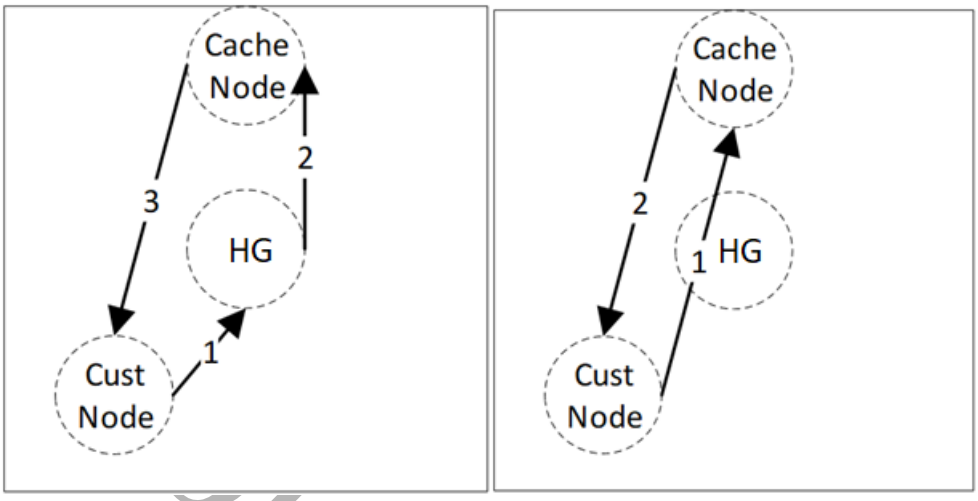
```
Procedure Initial (input : location_all_of_nodes;
output: HG, GG)
    Calculate nodes distance;
    For i= 1: all_of_nodes
        For k=1: all_of_nodes
            If i != k
                sent status_info from i to k
            EndIf
        EndFor
    EndFor
    Find number_of_grid
    Find set_node_of_grid
    For j=1: number_of_grid
        For m= 1: set_node_in_grid j
            For n= 1: set_node_in_grid j
                If m != n
                    sent status_info from m to n
                EndIf
        EndFor
    EndFor
    Pick HG %for every grid
    Pick GG %for every grid
EndProcedure;

Maintenance State
For every 60s
    For c = 1: number_of_nodes
        Sent info_updates to all of nodes
      Initial (location_all_of_nodes; HG, GG)
        Each node updates HG and GG
    EndFor
EndFor
```



Perutean Intra-Grid

Komunikasi intra-grid terjadi ketika pelanggan dan *node* yang di-cache berada di area grid yang sama. Pada saat sebuah *node* menginginkan konten maka ia akan mengirimkan paket *interest*. *Node* akan melakukan checkin pada FIB jika terdapat tujuan *node* destinasi terkait konten. Jika FIB memiliki database tersebut maka *node* akan membangun koneksi langsung menuju *node* tujuan. Jika tidak terdapat database maka *node* akan mengirimkan *interest* paket ke HG dan CH akan

meneruskan ke *node* tujuan pada internal grid, sebagaimana dapat dilihat pada Gambar IV.7 dan tahapan prosesnya dapat dilihat pada Algoritma 2:

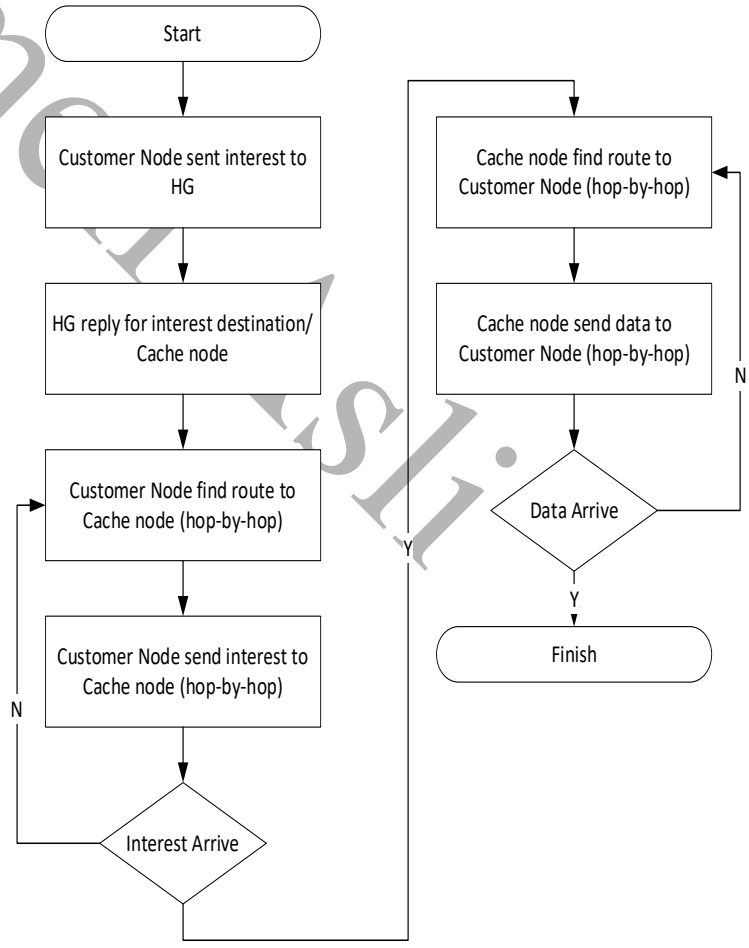


(a) (b)

Gambar IV.7 Perutean *Intra-grid*; (a). Via informasi HG; (b). Koneksi langsung

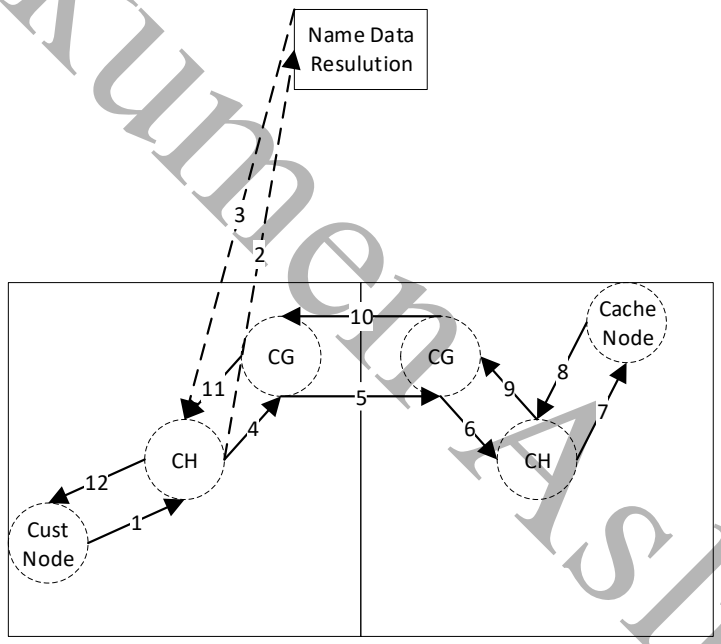
Algorithm 2: Intra-grid routing

```
Initial (location_all_of_nodes; HG, GG);
If consumer_node has location_data
    While not_reach location_data
        Geo_algorithm (input: neighbor_location;
output:next_hop)
        Sent interest_packet to next_hop
    EndWhile
    Next_hop sent reply_packet to consumer_node
Else
    While not_reach HG_of_location_data_grid
        Geo_algorithm (input: neighbor_location;
output: HG)
        Sent interest_packet to HG
    EndWhile
    HG forward to consumer_node
EndIf
```



Perutean Inter-Grid

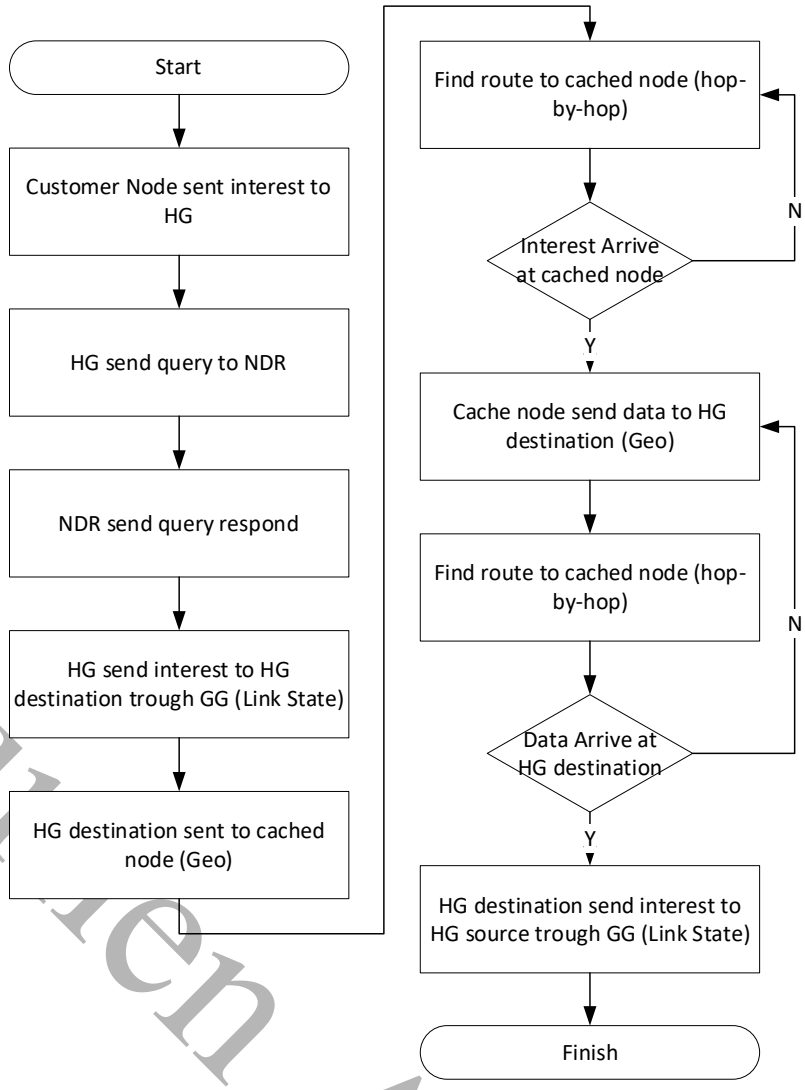
Jika konten yang diinginkan belum pernah di-cache pada intra-grid, maka pencarian melibatkan Named Data Resolution (NDR) (Hong et al., 2020), HG, dan GG antar grid. Pertama, *node* pelanggan mengirimkan *Interest* ke HG. HG mengecek FIB apakah memiliki database kueri terkait alamat konten yang diinginkan (diasumsikan belum dicatat di PIT). Jika tidak ada kueri dari konten yang diinginkan, HG mengirimkan permintaan kueri ke NDR, kemudian NDR mengirimkan kueri ke HG. Dengan grid id kisi kueri dari *node* yang di-cache diketahui, HG mengirim interest ke HG dari grid tujuan melalui GG. Selanjutnya, ketika paket interest mencapai HG tujuan, HG merelay paket interest ke *node* cache. *Node* cache kemudian membalas dengan data/konten ke *node* pelanggan, seperti yang dapat dilihat pada Gambar IV.8 dan disajikan pada Algoritma 3.



Gambar IV.8 Perutean *Inter-grid*

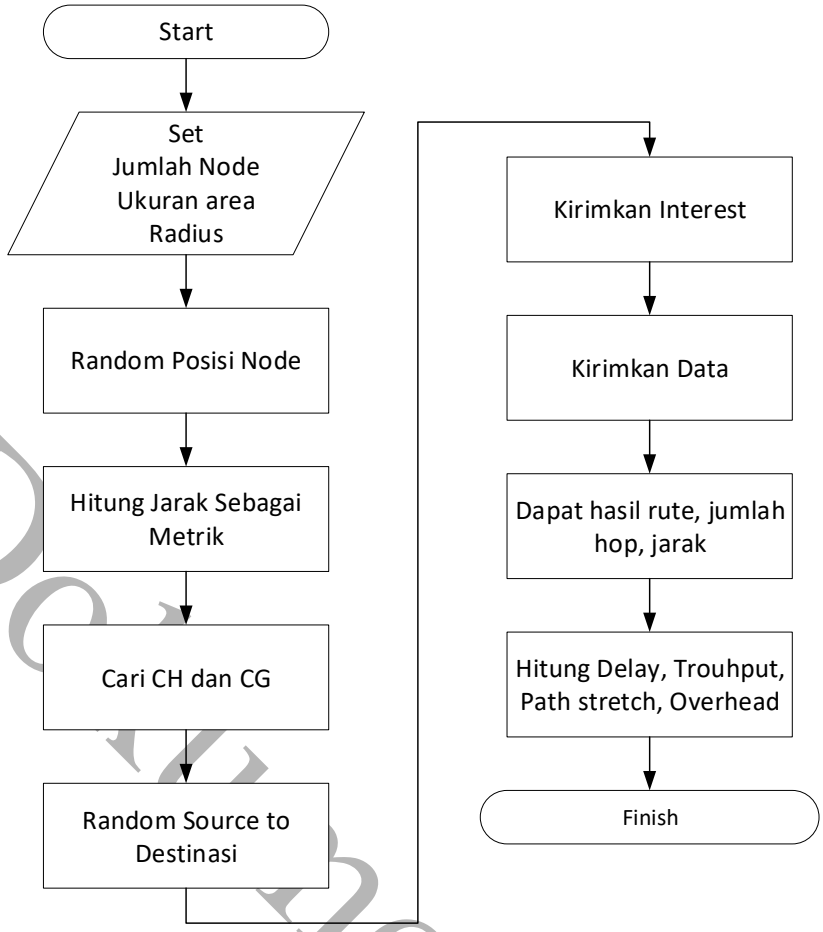
Algorithm 3: Inter-grid routing

```
Initial (location_all_of_nodes; HG, GG);
Consumer_node sent interest_packet to HG
If HG has query
    HG sent interest_packet to GG
else
    HG sent a query req to NDR
    NDR sent a query to HG
EndIf

X=1; Y=2;
While not_reach HG_dest & X< jum_HG
    HG_X sent interest_packet to HG_Y
    X=Y;       Y=+1;
EndWhile

While not_reach_cache_node
    Geo_algorithm (input:
neighbor_location_grid_CH_dest; output:
next_hop)
    Sent interest_packet to next_hop
EndWhile
```



IV.3 Analisa GSNR

Untuk menganalisis kinerja algoritma perutean GSNR, kami melakukan pengujian simulasi menggunakan MATLAB. Simulasi ini membandingkan GSNR dengan perutean NDN, NLSR, dan Perutean Geografis yang banyak digunakan berdasarkan pencarian serakah (GEO). Kami menghasilkan posisi acak *node* dan probabilitas acak koneksi *node*. Kami menyimulasikan peningkatan area jaringan yang juga meniru peningkatan jumlah *node* yang terlibat. Parameter simulasi digambarkan pada Tabel IV.1. Flow simulasi percobaan dapat dilihat pada



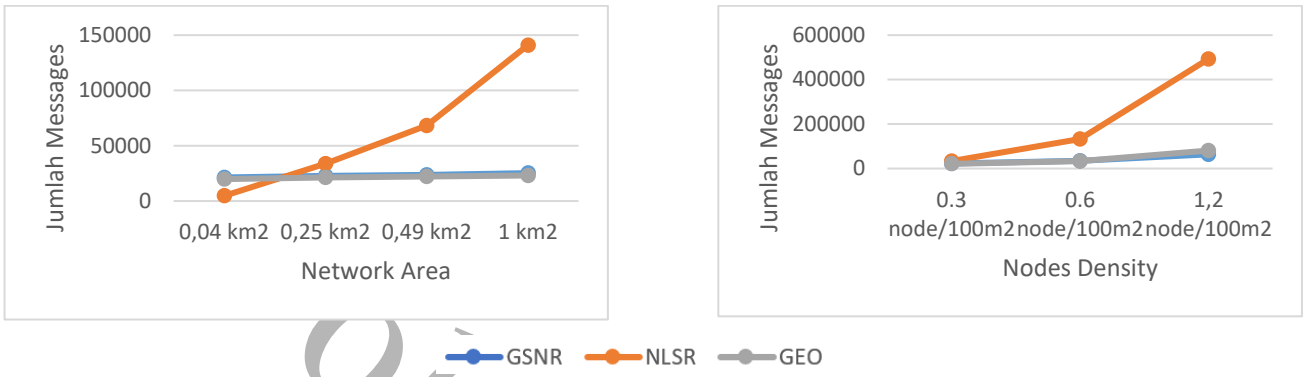
Gambar IV.9 Flow Simulasi GSNR

Tabel IV.1 Parameter Simulasi.

Parameters	**Value**
Data Packet Size	64 KB
Interest Packet Size	452 B
Network Area	0,04 ; 0,25; 0,49; 1 km²
*Node*s Density	0,3; 0,6; 1,2 *Node*s/100 m²
Node Coverage (r)	50 m
Grid/*Cluster* Size	100 m x 100 m
Link Bandwidth	54 Mbps

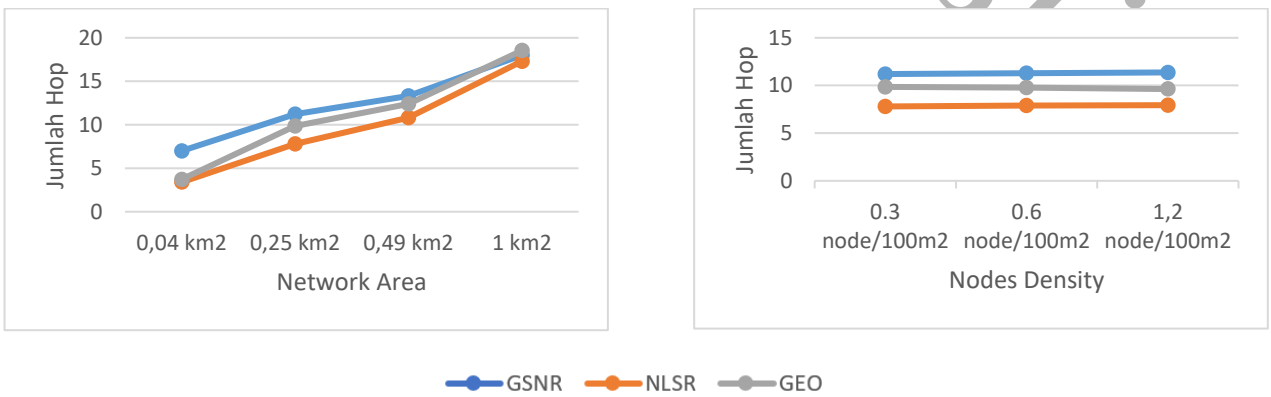
Dari pengukuran *overhead* pesan, umumnya *overhead* meningkat sebanding dengan peningkatan area dan *node*. *Overhead* pesan meningkat karena lebih banyak persinyalan diperlukan untuk mencakup situs dan semua *node*. Rata-rata kenaikan *overhead* pesan adalah 6,62 % untuk GSNR, 273 % untuk NLSR, dan 5,25 % untuk GEO, digambarkan pada Gambar IV.10. *Overhead* pesan juga meningkat karena

peningkatan kepadatan *node*. Lebih banyak *node* membutuhkan lebih banyak pesan untuk menjalankan mekanisme perutean dari biasanya. Rata-rata kenaikan *overhead* pesan adalah 81 % untuk GSNR, 282 % untuk NLSR, dan 100,3 % untuk GEO.



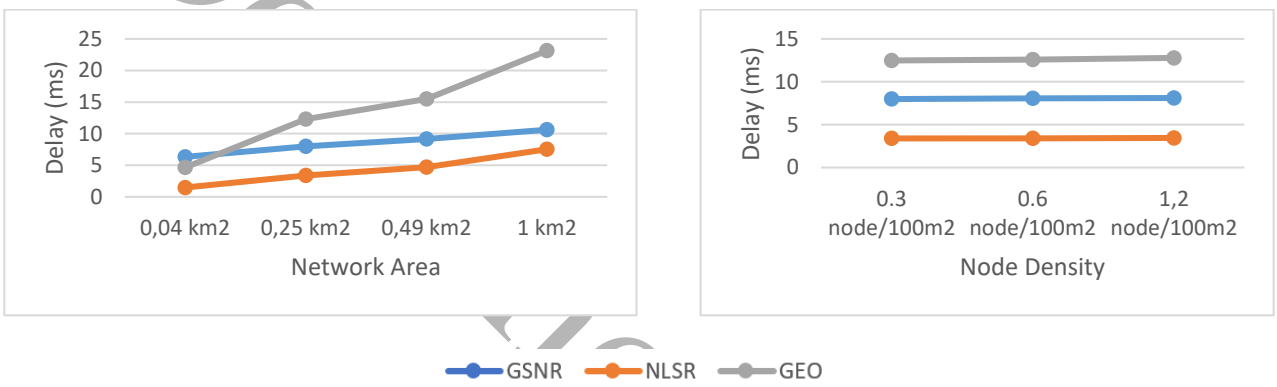
Gambar IV.10 Perbandingan *Overhead* Protokol Perutean

Peningkatan pesan *overhead* terjadi karena lebih banyak *node* yang dicakup oleh protokol perutean dalam pencarian rute di area yang lebih luas. Peningkatan relatif stabil pada protokol perutean GSNR dan GEO. Hal itu bisa terjadi karena kedua perutean protokol menempati perutean yang proaktif, sehingga perutean protokol hanya mencari rute yang diperlukan saja. *Overhead* perutean meningkat secara signifikan karena peningkatan kepadatan *node*. Namun, GSNR menjadi lebih mudah beradaptasi dengan peningkatan *overhead* terendah daripada protokol perutean lainnya.



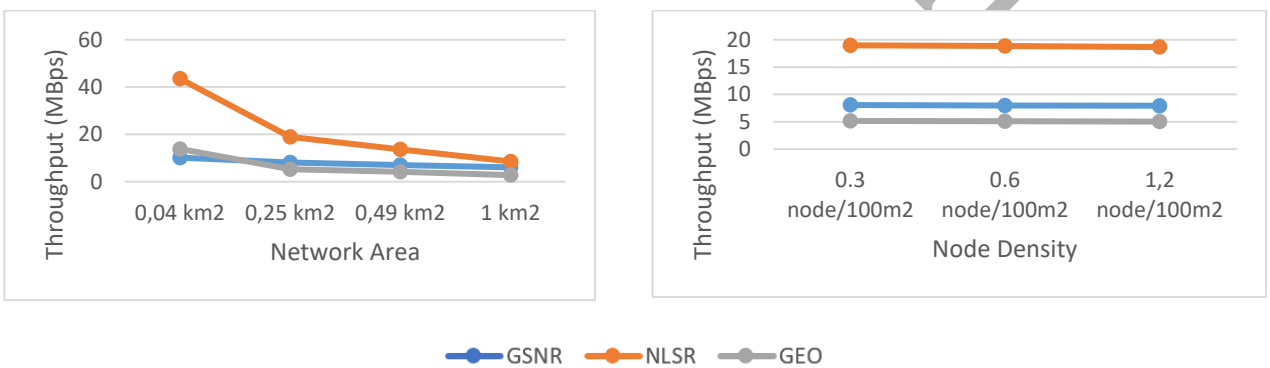
Gambar IV.11 Perbandingan Path-stretch Protokol Perutean

Dari pengukuran path-stretch, secara umum path-stretch bertambah sebanding dengan pertambahan luas. Namun demikian, jalur-peregangan tidak meningkat secara signifikan karena peningkatan kepadatan *node*. Path-stretch meningkat karena koneksi rute yang lebih panjang karena area yang lebih luas, yang membutuhkan lebih banyak lompatan dari sumber ke tujuan. Rata-rata pertambahan path stretch adalah 38,02% untuk GSNR, 76,02% untuk NLSR, dan 79,91% untuk GEO, digambarkan pada Gambar IV.10. GSNR memiliki pertambahan bentangan jalan terkecil seiring dengan perluasan wilayah. Artinya GSNR dapat diimplementasikan dalam berbagai ukuran area/jaringan tanpa mempengaruhi mekanisme perutean untuk mencari jalur terpendek (path stretch).



Gambar IV.12 Perbandingan Delay Protokol Perutean

Delay meningkat karena jarak rata-rata dan lompatan dari pertumbuhan sumber-tujuan. Rata-rata pertambahan delay adalah 18,92% untuk GSNR, 76,02% untuk NLSR, dan 79,90% untuk GEO, digambarkan pada Gambar IV.11. GSNR memiliki rata-rata pertambahan delay terkecil seiring dengan perluasan area. GSNR dapat mengejar kinerja penundaan NLSR sebagai perutean proaktif penuh.



Gambar IV.13 Perbandingan *Throughput* Protokol Perutean

Throughput menurun secara proporsional dengan peningkatan area tetapi cenderung stabil karena peningkatan kepadatan *node*. Rata-rata penurunan throughput adalah 15,77% untuk GSNR, 40,59% untuk NLSR, dan 38,60% untuk GEO, yang digambarkan pada Gambar IV.12. Untuk pertama kalinya, throughput GSNR kurang dari NLSR atau bahkan GEO. Namun, ketika perutean proaktif memiliki peran yang lebih menonjol pada jarak yang lebih jauh, GSNR memberikan performa yang lebih baik.

IV.4 Kompleksitas Algoritma

Secara umum, selalu terdapat banyak cara untuk menyelesaikan suatu masalah pada sebuah keilmuan, khususnya bidang *computer science*. Oleh karena hal tersebut, sangatlah penting terdapat metode untuk membandingkan solusi-solusi tersebut untuk mencari solusi yang paling optimal.

Kompleksitas adalah suatu ukuran yang menggambarkan berapa banyak waktu atau ruang yang dibutuhkan oleh suatu algoritma untuk menyelesaikan masalah. Notasi Big O adalah sebuah notasi matematika yang digunakan untuk menunjukkan kompleksitas algoritma dalam tata letak terbaik (best case), tata letak terburuk (worst case), dan tata letak rata-rata (average case). Notasi Big O memperkirakan batas atas performa algoritma, sehingga memungkinkan untuk membandingkan kinerja berbagai algoritma.

Contoh notasi Big O:

- $O(1)$ berarti bahwa kompleksitas algoritma tersebut tetap sama terlepas dari ukuran masalah.
- $O(n)$ berarti bahwa kompleksitas algoritma tersebut berbanding lurus dengan ukuran masalah.
- $O(n^2)$ berarti bahwa kompleksitas algoritma tersebut berbanding lurus dengan kuadrat ukuran masalah.

GSNR merupakan protokol perutean hibrid yaitu perutean proaktif berbasis algoritma Link State (LS) dan perutean reaktif berbasis algoritma *greedy* geografis (GR). Seperti dijelaskan sebelumnya algoritma LS akan berjalan untuk komunikasi *inter cluster* sedangkan algoritma GR akan berjalan pada komunikasi *intra cluster*.

Berikut adalah *pseudocode* untuk algoritma LS ketika mencari jalur untuk komunikasi inter *cluster* dimana melibatkan masing-masing *cluster* head dan *cluster gateway*. Secara umum algoritma link state memiliki nilai notasi big O adala $O(n^2)$. Artinya jika ada *node* sejumlah n di jaringan akan menyusun perutean, pertama akan status *node* tetangga langsung kemudian datanya disebar untuk mencari kembali status tetangga lainnya sehingga kompleksitasnya menjadi n^2

```
INITIALIZE network_topology, cluster_head, cluster_gateway, link_costs, LSPs, routing_tables

FOR cluster_head & cluster_gateway in network_topology DO
  link_costs[node] = calculate_link_costs(node)
END FOR

FOR cluster_head & cluster_gateway in network_topology DO
  LSAs[node] = create_LSDB(node, link_costs[node])
  broadcast(LSAs[node])
END FOR

FOR cluster_head & cluster_gateway in network_topology DO
  FOR each received_LSA in LSAs DO
    update_network_topology(network_topology, received_LSA)
  END FOR
END FOR

WHILE true DO
  FOR cluster_head & cluster_gateway in network_topology DO
    link_costs[node] = calculate_link_costs(node)
  END FOR

  FOR cluster_head & cluster_gateway in network_topology DO
    LSAs[node] = create_LSDB(node, link_costs[node])
```

```
    broadcast(LSAs[node])
  END FOR

  FOR cluster_head & cluster_gateway in network_topology DO
    FOR each received_LSA in LSAs DO
      update_network_topology(network_topology, received_LSA)
    END FOR
  END FOR

  FOR cluster_head & cluster_gateway in network_topology DO
    routing_tables[node] = calculate_shortest_path(node, network_topology)
  END FOR

FOR each data_transmission DO
  source_node = data_transmission.source
  destination_node = data_transmission.destination

  IF routing_tables[source_node] contains destination_node THEN
    path = routing_tables[source_node][destination_node]
    send_data(data_transmission, path)
  ELSE
    handle_routing_failure(data_transmission)
  END IF
END FOR
```

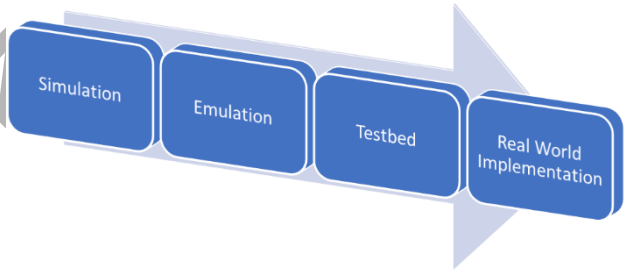
Selanjutnya, berikut adalah pseudocode algoritma GR dengan mengumpulkan informasi *node* tetangganya dan memilih mana yang terdekat menuju *node* tujuan. Pencarian dilakukan hop by hop dimana untuk pencarian tersebut GR mengeliminasi pencarian selanjutnya untuk *node* tetangga yang tidak terpilih. Sehingga dalam penerapannya algoritma GR membagi kelompok *node* tetangga sebagai next hop dan eliminated neighbour shingga dari n masukan akan diperkecil hingga setengahnya sehingga jumlah maksimal iterasinya menjadi n, dimana n adalah jumlah data. Sehingga kompleksitas notasi Big O untuk GR adalah $O(n)$.

```
# Initialize node coordinates
for each node in all nodes
    node.coordinate = get_coordinate(node)

function greedy_geographic_routing(source, destination)
    current_node = source
    while current_node ≠ destination
        next_node = find_nearest_neighbor(current_node, destination)
        if next_node is not reachable
            return failure
        current_node = next_node
    return success

function find_nearest_neighbor(current_node, destination)
    nearest_neighbor = null
    min_distance = infinity
    for each neighbor in current_node's neighbors
        distance = calculate_distance(current_node.coordinate, neighbor.coordinate,
destination.coordinate)
        if distance < min_distance
            min_distance = distance
            nearest_neighbor = neighbor
    return nearest_neighbor
```

Pada algoritma GSNR mekanisme *clustering* menyederhanakan proses atau jumlah inputan n menjadi lebih sedikit karena ketetanggan hanya melibatkan *cluster* head dan *cluster gateway* sehingga solusi masalahnya sebanding dengan nilai inputan $\sqrt{n}$ sehingga nilai big O adalah $O(n + \sqrt{n})$ dan dapat disederhanakan menjadi $O(n)$. Dalam hal ini, kompleksitas GSNR lebih rendah daripada LS dan setara dengan GR, sebesar $O(n)$, dimana n adalah jumlah total node dalam sistem. Namun GSNR memiliki performa sistem lebih bai daripada GR yang telah dibuktikan pada sub bab IV.3

Bab V Kerangka kerja NDN *Testbed* Pada Virtual Machine

Banyak penelitian dilakukan untuk mencoba mengimplementasikan NDN di dunia nyata. Secara umum tahapan penelitian yang dilakukan adalah simulasi, emulasi, dan uji coba (testbed dan implementasi), sebagaimana dapat dilihat pada Gambar V.1. Pada tahap awal simulasi pada penelitian NDN biasanya menggunakan NDNSim (Mastorakis et al., 2017). NDNSim dikembangkan dari kerangka simulasi NS-3 dan digunakan secara luas oleh peneliti. Penelitian berbasis NDN IoT (Amadeo et al., 2019), (Amadeo et al., 2020) menggunakan simulasi NDNSim untuk memverifikasi algoritma yang diusulkan. NDNSim menggunakan kerangka kerja NS-3 dengan memanfaatkan fitur, utilitas dan *helpers* yang ada.



Gambar V.1 NDN real-world implementation steps

Namun demikian, setiap rilis NDN *Forwarding Daemon* (NFD) dan ndn-cxx harus dilakukan terintegrasi secara manual dengan NDNSim. NFD berfungsi sebagai penerus jaringan, dan ndn-cxx adalah pustaka C++ untuk mengimplementasikan NDN. Komponen tersebut adalah bagian penting dalam membuat *node* NDN. Terlebih lagi, beberapa elemen dalam NDNSim terkadang menunjukkan anomali saat dijalankan karena beberapa penyederhanaan dibuat dalam simulasi. Oleh karena itu, kami perlu memastikan bahwa pengujian unit yang ada dan yang baru ditambahkan berjalan secara konsisten

Emulasi adalah langkah selanjutnya dalam penelitian NDN. Dalam (Budiana et al., 2021) menggunakan Mini-NDN untuk menyelidiki dampak contentstore terhadap skema penggantian cache. Pada penelitian (Guo et al., 2021) mengusulkan skema NOLSR untuk protokol perutean NDN-MANET menggunakan emulator Mini-NDN. Emulasi bermaksud untuk mendekati mekanisme perangkat keras dan sistem

operasi lebih dekat dengan proses dunia nyata. Mayoritas peneliti NDN menggunakan Mini-NDN untuk meniru proses NDN. Mini-NDN adalah alat emulasi jaringan ringan yang dibangun di atas Mininet. Ini memungkinkan pengujian, eksperimen, dan penelitian NDN untuk mencapai skalabilitas. Pustaka NDN, NFD, NLSR, dan alat yang dirilis oleh proyek NDN untuk meniru jaringan NDN. Mini-NDN berjalan di buruh pelabuhan sebagai lingkungan Virtual Machine (VM) tingkat kontainer. Oleh karena itu Mini-NDN dapat meniru *node* dan jaringan NDN lengkap pada satu sistem. Setiap pengiriman paket dan proses di Mini-NDN dapat dianalisis menggunakan alat penangkap paket (Wireshark atau Tcpdump) dan file logging proses (NFD dan NLSR). Namun, Mini-NDN tidak memiliki proses isolasi, dan alokasi sumber daya CPU/RAM untuk setiap *node*.

Langkah terdekat untuk implementasi dunia nyata adalah *testbed*. Pada (Saxena and Raychoudhury, 2019), Divya Saxena dan Vaskar Raychoudhury menyimulasikan dan membuat *testbed* Smart Healthcare berbasis NDN menggunakan perangkat IoT. L. Gameiro, C. Senna, dan M. Luís dalam (Gameiro et al., 2022) mengimplementasikan Intelligent Transportation System berbasis NDN menggunakan Raspbery Pi dan Nvidia Jetson. Dalam *testbed* biasanya peneliti menggunakan resource khusus untuk mengimplementasikan *node* NDN pada skala lab (Friyanto et al., 2019). Skala khusus berarti peneliti harus mengalokasikan sumber daya nyata untuk menjalankan *node* NDN. Dengan alokasi sumber daya khusus, setiap proses dapat dipantau: penggunaan CPU, penggunaan RAM, bahkan konsumsi energi per *node*. Akibatnya, metode ini menjadi kurang hemat biaya daripada yang lain.

Untuk mengatasi masalah tersebut, membangun lingkungan dengan alam yang lebih dekat dengan dunia implementasi memiliki kemudahan penggunaan, dan diperlukan observasi. Karena kebutuhan ini, kami mengusulkan kerangka kerja yang menggabungkan keunggulan emulasi dan *testbed*. Kerangka kombinasi ini mengumpulkan tujuan penelitian NDN sedekat mungkin dengan implementasi dunia nyata. Kami mengusulkan proses emulasi berbasis UNetlab menggunakan VM berbasis Qemu untuk pendekatan implementasi yang lebih mudah, pemantauan

yang komprehensif, dan efektivitas dari sisi biaya. Hasil penelitian ini menunjukkan bagaimana kerangka kerja yang diusulkan dapat memantau parameter QoS, penggunaan CPU, dan penggunaan RAM. Penggunaan CPU dan RAM tidak mungkin untuk diperiksa dan dianalisis jika kita hanya menggunakan NDNSim atau bahkan Mini-NDN. Kami percaya bahwa kerangka kerja ini dapat membawa peneliti selangkah lebih dekat untuk mengimplementasikan jaringan dan teknologi NDN.

V.1 NDN Virtualization

Virtualisasi sangat penting untuk memodelkan *node* NDN karena kami dapat meningkatkan fleksibilitas sistem terdistribusi, isolasi proses, dan toleransi kesalahan. Isolasi proses sangat penting karena setiap *node* berjalan secara independen dengan sumber daya perangkat keras khusus di lingkungan nyata. Tanpa isolasi, tidak akan mudah untuk memantau kinerja *node* NDN.

Sebelumnya, IBM menginisialisasi teknologi virtualisasi pada tahun 1973, dan melalui waktu virtualisasi menjadi sangat populer dan digunakan secara luas. Saat ini level/jenis dan penerapan virtualisasi sudah semakin bervariasi. Mulai dari virtualisasi penuh awal dan para-virtualisasi berkembang menjadi virtualisasi tingkat sistem operasi yang dapat diskalakan.

Implementasi virtualisasi misalnya digunakan untuk mengoperasikan beberapa Virtual Machine (VM) pada satu mesin fisik atau *server*. Mekanisme ini akan meningkatkan penggunaan sumber daya *server* dalam hal CPU, RAM, dan energi. Satu *server* khusus biasanya hanya menggunakan beberapa sumber daya perangkat keras yang tersedia. Penggunaan VM juga meningkatkan fleksibilitas perusahaan untuk menjalankan operasi *server*. Menambahkan sumber daya, pencadangan sumber daya, migrasi, dan operasi *server* lainnya sangatlah mudah. Manfaat ini, pada akhirnya, akan memberikan kehandalan operasi perusahaan yang lebih baik.

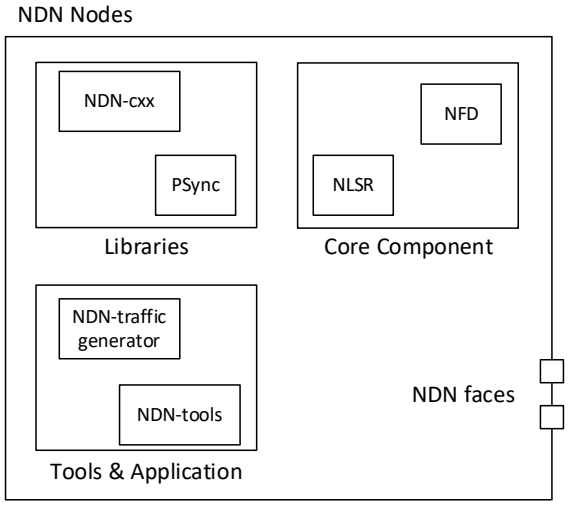
Virtualisasi juga diadopsi dalam dunia komunikasi data. Ada istilah yang disebut Virtualization Network Function (VNF). VNF, virtualisasi jaringan dan fungsi lainnya, seperti keamanan jaringan, dari perangkat keras khusus. Baik penyedia

layanan jaringan dan perusahaan telah banyak menggunakan metode VNF ini dalam virtualisasi *router*, switch, firewall, penyeimbang beban, *server* DNS, dan mekanisme caching.

Dalam makalah ini, kami mengusulkan kerangka kerja berdasarkan mekanisme virtualisasi dalam penelitian NDN sebagai langkah tambahan untuk implementasi di dunia nyata. Pada langkah ini, peneliti memvirtualisasikan fungsi NDN sebelum masuk ke langkah *testbed*. Pada langkah virtualisasi NDN, mereka memvirtualisasikan fungsi NDN dalam perangkat lunak, sehingga algoritma atau fungsi baru yang diusulkan oleh peneliti dapat digunakan sebagai VM atau *container*. Dengan batu loncatan ini, penelitian NDN akan lebih cepat dan efisien. Selanjutnya, dalam virtualisasi NDN, peneliti dapat memantau kinerja sistem yang diatur sedekat mungkin dengan sumber daya perangkat keras dari sistem kandidat *testbed*.

V.1.1 Named Data Networking *Node*s

Node NDN dibangun di atas pustaka inti/berbasis kode yang disebut NDN-cxx. NDN-cxx berkolaborasi dengan komponen inti (NFD & NLSR), alat NDN, dan aplikasi (NDN traffic generator), seperti yang digambarkan pada Gambar V.2. NDN-cxx digunakan sebagai pustaka kode NDN yang pertama kali dibuat dan digunakan untuk menulis berbagai aplikasi NDN. Fungsi komponen inti adalah komponen NDN yang wajib untuk menjalankan fungsi *node* NDN, seperti perutean, caching, dan *forwarding*. Sedangkan alat dan aplikasi NDN digunakan sebagai ekstensi untuk percobaan. Wajah *node* NDN mirip dengan antarmuka di *router* yang terhubung ke *node*, konsumen, atau produsen NDN lainnya.



Gambar V.2 Komponen *node* NDN

Ada dua jenis paket NDN, bunga dan data. Konsumen menggunakan paket bunga untuk meminta konten, sedangkan paket data dikirim ke konsumen sebagai tanggapan atas paket bunga. Padahal fungsinya berbeda, kepentingan dan paket data menggunakan nama sebagai pengenal konten. Setiap *node* dalam jaringan NDN juga memiliki nama, sehingga mekanisme perutean menggunakan nama.

V.1.2 UNetlab

UNetlab adalah platform emulator multi-vendor dan multi-pengguna gratis untuk membuat dan memodelkan berbagai fungsi dan topologi jaringan. Ada yang mengatakan itu meniru, tetapi metode meniru fungsi jaringan memiliki level yang lebih tinggi daripada di Mini-NDN. UNetlab dapat memvirtualisasikan *router*, switch, firewall, pelanggan, dan produsen. Emulator memiliki dukungan luas untuk banyak jenis peralatan dari berbagai vendor, seperti Cisco, Citrix, PaloAlto, Juniper, VMWare, Windows *Server*, Aruba, dan Fortinet. Karena NDN belum distandarisasi, maka tidak terdaftar sebagai fungsi yang didukung. Meski begitu, kita dapat membangun fungsi NDN di atas fungsi sistem operasi yang didukung seperti Ubuntu atau Sistem Operasi berbasis UNIX lainnya.

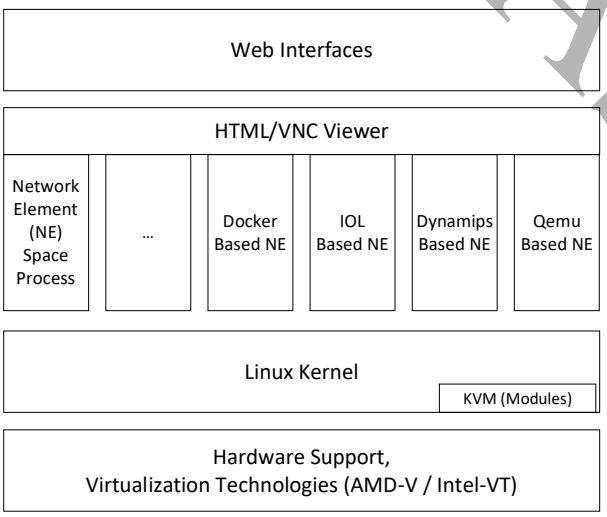
EVE-NG (“EVE-NG Site,” n.d.) dan PNETLab (“PNETLab : Lab is Simple Site,” n.d.) adalah bagian dari pengembangan emulator UNetlab. Mengutamakan kemudahan penggunaan, user interface yang friendly, dan menyediakan opsi virtualisasi hingga level container inilah yang menjadi preferensi kami. EVE-NG

dan PNETlab memiliki antarmuka dan fitur pengguna yang hampir sama. Namun dengan beberapa pertimbangan seperti jumlah lisensi gratis, contoh kasus yang pernah dilakukan, dan dukungan media sosial, dalam penelitian ini kami memutuskan untuk menggunakan PNETLab.

Node NDN dibuat melalui virtualisasi Qemu berbasis PNETLab dan diinstal pada OS Bodhi Linux 32bit kecil. Kami menggunakan Bodhi Linux karena sumber daya perangkat kerasnya yang tidak menuntut. Persyaratan minimum Bodhi Linux adalah prosesor 500MHz, RAM 512MB, dan ruang drive 5GB. Karena Bodhi Linux dibangun berdasarkan Ubuntu LTS, kami tidak menemukan banyak kesulitan dalam menginstal komponen NDN ke OS ini.

V.1.3 UNetlab Network Element & QEMU

UNetlab mendukung empat jenis lingkungan virtualisasi untuk proses ruang Elemen Jaringan. Mereka adalah Docker, IOL, Dynamips, dan Qemu, seperti yang digambarkan pada Gambar. V.3. Semuanya berjalan pada dukungan perangkat keras (AMD-V atau Intel-VT) tergantung pada spesifikasi perangkat keras dan modul KVM kernel Linux. Setiap perangkat keras yang kompatibel memiliki persyaratan khusus untuk ditiru. Selanjutnya, Pilihan lingkungan virtualisasi juga akan mempengaruhi kinerja perangkat keras yang ditiru.



Gambar V.3 Struktur UNetlab

Docker (“Home - Docker,” n.d.) adalah level kontainer dari lingkungan virtualisasi. Docker terkenal dan banyak digunakan karena fleksibilitas dan manajemen sumber dayanya yang adaptif. Di UNetlab, perangkat/fitur yang didukung sangat minim. Docker untuk Wireshark, chrome, Ubuntu, ansible, dan Cisco NSO paling banyak digunakan untuk pengujian, pengambilan, dan orkestrasi perangkat.

Sistem Operasi Internetwork over Linux (IOL) adalah emulasi yang didedikasikan untuk perangkat Cisco IOS Systems. IOL dapat mendukung *router* Cisco, switch r, firewall, SD-WAN, dan *server* yang berjalan di sistem operasi Linux.

Dynamips (“GitHub - GNS3/dynamips: Dynamips development,” n.d.), seperti IOL, juga dibangun untuk meniru perangkat cisco. Dynamips banyak digunakan sebagai tolok ukur pengujian telekomunikasi eksperimental dan untuk persiapan ujian cisco. Didukung oleh komunitas GNS3, Dynamips git secara aktif mengembangkan sistem. Awalnya, Dynamips hanya mendukung *router* Cisco 7200; saat ini, dapat meniru rangkaian *router* Cisco lainnya, seperti keluarga *router* Cisco 3600, 3700, dan 2600.

QEMU adalah singkatan dari Quick Emulator, emulator dan virtualizer mesin generik dan *open-source*. Ada dua jenis level emulasi QEMU. Tipe pertama adalah emulasi Sistem. Dalam mode ini, QEMU meniru mode virtual seluruh mesin. Ini termasuk CPU, memori, dan perangkat yang ditiru yang menjalankan OS tamu. Tipe kedua adalah emulasi mode Pengguna. QEMU dapat memanggil executable Linux yang dikompilasi untuk arsitektur perangkat yang berbeda dalam mode ini. QEMU dapat memanfaatkan sumber daya sistem host untuk meluncurkan proses yang dikompilasi ke arsitektur berbeda dalam mode ini. Karena mekanisme yang rumit ini, beberapa fitur mungkin mengalami masalah kompatibilitas. Perintah QEMU untuk emulasi mode pengguna diberi nama qemu-target_architecture, misalnya, qemu-x86_64 untuk meniru CPU Intel 64-bit.

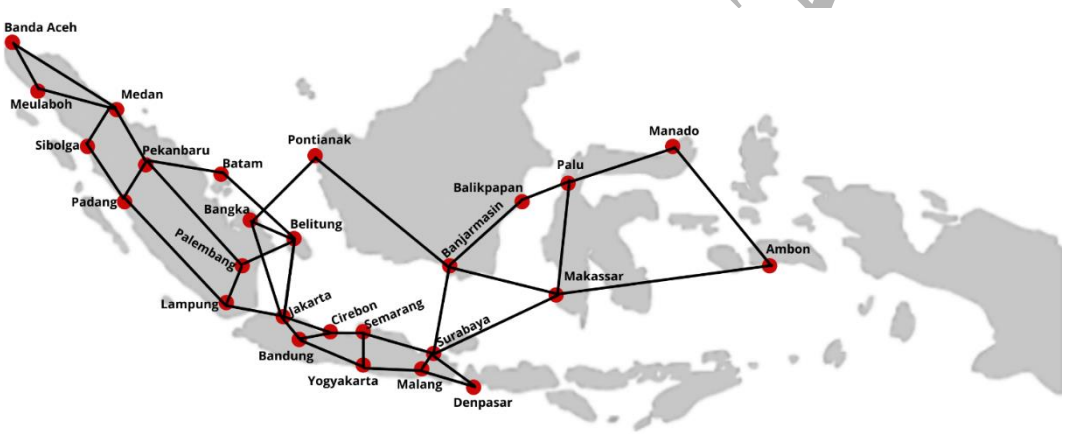
Untuk memvirtualisasikan *node* NDN di lingkungan UnetLab, kami menggunakan virtualisasi berbasis Qemu. Pemilihan Qemu karena implementasi NDN pada OS berbasis Linux (Bodhi dan karena fleksibilitas dukungan lingkungan virtualisasi Qemu.

V.2 System Model

Dalam penelitian ini, kami membuat beberapa studi kasus untuk menunjukkan bagaimana sistem yang diusulkan ini dapat membantu para peneliti NDN untuk percobaan intensif. Dalam penelitian kami, sistem NDN ini digunakan untuk menganalisis kinerja protokol perutean proaktif di Indonesia Digital Network (IDN) berbasis NDN.

V.2.1 IDN

IDN memainkan peran penting dalam strategi pembangunan Indonesia. Infrastruktur komunikasi data diyakini menjadi langkah penting untuk memperkuat posisi Indonesia dalam menghadapi tantangan perubahan global. Untuk mewujudkan visi tersebut, pemerintah dan pemangku kepentingan membangun Indonesia Digital Network (IDN) yang berfokus pada pengembangan dan penyediaan konektivitas internet untuk seluruh wilayah Indonesia. Infrastruktur telekomunikasi digital meliputi kabel serat optik dan terestrial di darat dan kabel bawah laut di laut, seperti terlihat pada Gambar V.4.



Gambar V.4 Model IDN

Untuk melihat seberapa intensif PNETLab (PL) dapat digunakan untuk pendekatan *testbed* jaringan berbasis NDN, kami membuat NDN berbasis IDN di lab virtual PNETLab dan kemudian membandingkan skenario dengan emulasi NDN berbasis Mini-NDN (MN) yang banyak digunakan. Banyak peneliti NDN mengadopsi emulasi Mini-NDN sebagai alat emulasi utama untuk NDN. Model IDN yang dibuat merupakan jaringan *backbone* yang membentang dari Banda Aceh hingga Ambon di timur Indonesia. Model ini melibatkan dua puluh enam *node*, yang terdiri dari enam belas kota utama dan sepuluh transit, di mana *node* NDN ditempatkan, seperti yang terlihat pada Gambar V.4. Dalam skenario, Jakarta berperan sebagai produsen, dan (Medan, Semarang, Pontianak, Makassar, Denpasar, Yogyakarta, Padang, dan Manado) sebagai konsumen.

V.2.2 Permintaan Konten

Dalam sistem ini, konsumen menghasilkan permintaan konten berdasarkan Zipf-Mandelbrot ("ndn-traffic-generator_modified," n.d.). Kami memodelkan lalu lintas dari permintaan konten konsumen menggunakan distribusi Zipf-Mandelbrot. Konten dengan popularitas peringkat teratas lebih cenderung diminta oleh konsumen daripada konten yang tidak populer. Probabilitas permintaan konsumen atas konten i dinyatakan dengan Persamaan. (V.1):

$$\boldsymbol{p(i)} = \frac{(\boldsymbol{i}+\boldsymbol{q})^{-\boldsymbol{\alpha}}}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{i}+\boldsymbol{q})^{-\boldsymbol{\alpha}}} \qquad \text{(V.1)}$$

Dimana:

p (i) = probabilitas permintaan konten i,

q = Faktor perebahan,

α = Faktor eksponensial,

N = Jumlah file atau konten.

V.2.3 *Network Cache Hit Ratio* (nCHR)

Dalam penelitian ini, istilah *'network cache hit'* menunjukkan jumlah permintaan konten yang dapat ditanggapi oleh *router* NDN perantara. *Router* NDN perantara berarti simpul NDN antara konsumen dan produsen. Jadi, satu permintaan dikatakan 'hit' ketika paket *interest* dapat dipenuhi oleh konten yang di cache oleh

node perantara sehingga paket interest tidak perlu diteruskan ke produser. Oleh karena itu, kemampuan *router* perantara NDN untuk memenuhi kepentingan dapat meningkatkan QoS tanpa membebani sumber daya produser. Sehingga cache hit ratio dihitung sebagai perbandingan antara jumlah *interest* konten yang berhasil dipenuhi oleh *cache node* NDN perantara (network cache hit) dibandingkan dengan jumlah total *interest* (*hit* dan *miss*). *Cache Hit Ratio* jaringan (nCHR) (Yovita et al., 2020) untuk skenario diperoleh dengan menggunakan rumus berikut:

$$nCHR = \frac{nH}{nH + nM} \quad \text{(V.2)}$$

dimana:

nH = jumalah hit pada node perantara,

nM = jumlah miss pada node perantara.

V.2.4 Utilisasi CPU dan RAM

Utilisasi CPU adalah parameter penggunaan CPU rata-rata selama periode pengamatan. Istilah persen pemanfaatan CPU diterapkan untuk bagian periode waktu di mana komponen CPU bekerja dan menjalankan pemrosesan, dibagi dengan jumlah total waktu pengamatan. Hasilnya dikalikan dengan 100% untuk mendapatkan persentase, dilambangkan dengan persamaan. (V.3).

Setidaknya terdapat 4 (empat) kelompok proses pada *Node* default NDN, Proses perutean, Proses *Caching*, Proses NDN *Forwarding*, dan Proses Background Lainnya; lihat Persamaan. (V.3) dan (V.4). Untuk pengukuran Utilisasi CPU NDN, uraikan di bagian berikutnya, dengan fokus pada persentase penggunaan CPU konsumen dan produsen NDN. Di konsumen NDN, ada proses penghasil lalu lintas, jadi Persamaan. (V.4) berkembang menjadi Persamaan. (V.5). Sementara itu, tidak ada proses penghasil lalu lintas di produsen NDN, dan tidak semua paket kepentingan sampai ke produsen. Karena mekanisme hit *cache*, Persamaan. (V.6), sehingga Persamaan. (V.4) berkembang menjadi Persamaan. (V.7)

$$U = \frac{\sum_{i=1}^{p} tCPUwork_i}{T} \times 100\% \quad \text{(V.3)}$$

$$U_{NDN} = \frac{tRP + tCP + tFP + tBP}{T} \times 100\% \quad \text{(V.4)}$$

$$U_{cons} = \frac{tRP + tCP + tFP + tBP + tTG}{T} \times 100\% \quad \text{(V.5)}$$

$$P(M) = 1 - P(H) \quad \text{(V.6)}$$

$$U_{prod} = \frac{tRP + tCP + \prod_{i=1}^{n} P_i(M)\, tFP + tBP}{T} \quad \text{(V.7)}$$

Dimana:

U= Utilisasi CPU

U_{NDN} = Utilisasi CPU NDN intermediate node

U_{cons} = Utilisasi CPU NDN consumer node

U_{prod} = Utilisasi CPU NDN producer node

T = Waktu pengamatan

tRP = Waktu Proses perutean NDN

tCP = Waktu Proses NDN *caching*

tFP = Waktu Proses NDN *forwarding*

tBP = Waktu Proses lainnya

tTG = Waktu Proses pembangkitan *traffic*

$\prod_{i=1}^{n} P_i(M)$*=Network cache missed*

n=Jumlah node sepenjang jalur

Berikut adalah arti dari rata-rata pemakaian CPU sebesar 75% selama waktu pengamatan. Hal ini artinya, CPU menempati 75% dari waktu pelaksanaan proses selama waktu observasi. Ketika utilisasi CPU mencapai tingkat penggunaan 100%, maka CPU bekerja sepanjang waktu pengamatan. Dalam hal ini, proses dapat membebani kapasitas CPU.

Memori juga dikenal sebagai penyimpanan utama, memori utama, penyimpanan utama, penyimpanan internal, memori utama, dan RAM (Random Access Memory); semua istilah ini biasa digunakan pada dunia per-komputeran. Memori

adalah bagian dari komputer yang menyimpan data dan instruksi untuk diproses oleh CPU. Meskipun terkait erat dengan CPU, memori merupakan bagian yang terpisah. Memori menyimpan instruksi program atau data hanya selama program yang bersangkutan sedang beroperasi. Data-data instruksi sebelum diproses ke CPU atau beberapa hasil pemrosesan sebelumnya disimpan atau diantrekan ke RAM sebelum ditransmisikan kembali ke CPU. Ketika jumlah instruksi meningkat di RAM maka utilitas CPU juga otomatis akan meningkat.

Waktu yang dibutuhkan dari instruksi mengantre dan mulai diproses di CPU biasa disebut sebagai waktu idle atau waktu I/O dimana persentase waktu idle dapat dihitung menggunakan persamaan V.8. Pada saat terdapat *n* instruksi berjalan pada RAM dan CPU maka utilisasi CPU dapat dihitung menggunakan persamaan V.9.

$$p = \frac{tidle}{T} \times 100\% \quad \text{(V.8)}$$

$$U = (1 - p^n) \times 100\% \quad \text{(V.9)}$$

Dimana:

U = Utilisasi CPU

T = Waktu pengamatan

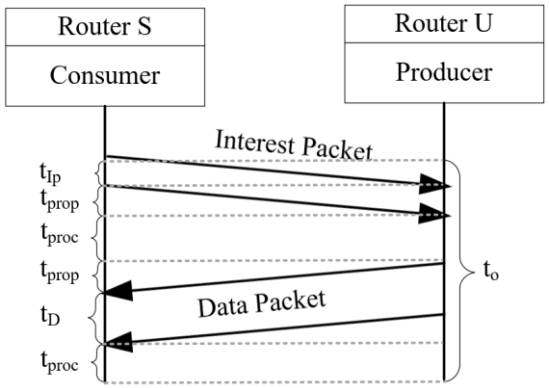
p = Persentase waktu idle / I/O

n = Jumlah instruksi pada RAM dan CPU

V.2.5 Throughput

Throughput didefinisikan sebagai jumlah rata-rata bit data yang ditransmisikan dalam satuan waktu. Transmisi paket data NDN digambarkan pada Gambar V.5. Komunikasi data dimulai ketika konsumen mengirimkan paket *interest* untuk konten yang diinginkan. Melibatkan beberapa mekanisme dan bisa dikatakan selesai pada saat konsumen menerima semua paket data dari produsen. Total waktu yang dibutuhkan untuk proses itu dapat dituliskan sebagai Persamaan. (V.8). Seluruh data yang diinginkan dapat berupa file berukuran besar yang dilambangkan dengan D Byte, seperti multimedia, gambar, atau musik. Dengan demikian, NDN membagi file D menjadi beberapa paket data c chunk dan mengirimkannya

sebanyak i kali. Dengan kondisi kanal non-blocking *node* NDN memiliki sumber daya cukup untuk pemrosesan paket data sehingga throughput T dapat menjadi proporsional dengan ukuran paket PS, seperti diuraikan pada Persamaan. (V.10)-(V.15).



Gambar V.5 Proses Komunikasi *Router* NDN

$$t_o = 2tprop + 2tproc + t_{Ip} + t_D \quad \text{(V.10)}$$

$$T = \frac{D}{\sum_{i=1}^{c} t_i} \quad \text{(V.11)}$$

$$T = \frac{D}{\sum_{i=1}^{c}(2tprop + 2tproc + t_{Ip} + t_D)_i} \quad \text{(V.12)}$$

$$\sum_{i=1}^{c} t_i \propto ct_i = \frac{D}{PS} t_i \quad \text{(V.13)}$$

$$T \propto \frac{D}{D/PS(2tprop + 2tproc + t_{Ip} + t_D)} \quad \text{(V.14)}$$

$$T \propto \frac{PS}{(2tprop + 2tproc + t_{Ip} + t_D)} \quad \text{(V.15)}$$

Dimana:

t_o	= Waktu transmisi(s),
$tprop$	= Waktu propagasi (s),
$tproc$	= Waktu proses paket (s)
t_{Ip}	= Waktu transmisi paket *interest* (s)
t_D	= Waktu transmisi data paket (s)
D	= Ukuran data paket (B)
PS	= Ukuran paket (B)

V.2.6 Parameter Lainnya

Detail parameter studi kasus PL ditunjukkan pada Table V.1.

Tabel V.1 Parameter Studi Kasus

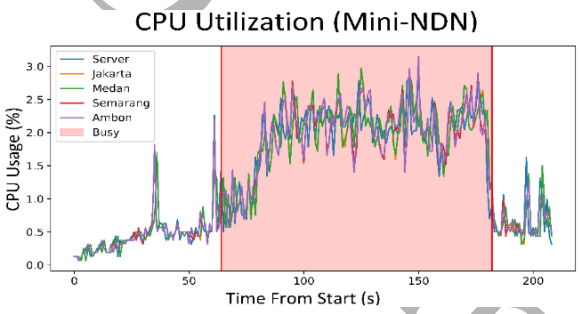
Parameter	**Nilai**
Jumlah *node* NDN	26
Jumlah produser	1 (Jakarta)
CPU *core*, *node* NDN	1
RAM *node* NDN	2048 MB
Jumlah konsumer	2, 5, 8 Konsumer
Ukuran paket	1024, 4096, 8192
Interest rate	100 int/s
Exp Faktor	0,8
Flattened faktor	3
Ukuran cache	30 paket
Link bandwidth	10, 100 Mbps
Skenario	26 NDN *node* virtualization

V.3 Evaluasi Performa

Dengan menggunakan kerangka kerja ini, kita dapat mengevaluasi parameter performa jaringan dan performa perangkat keras *node* NDN. Kinerja skenario NDN-IDN menggunakan PL dibandingkan dengan skenario NDN-IDN menggunakan MiniNDN (MN), karena emulasi berbasis NDN banyak digunakan pada penelitian NDN dan menjadi *best-practice*. Dalam banyak penelitian yang menggunakan emulasi MN, para peneliti tidak menganalisis penggunaan CPU dan RAM dari skenario tersebut. Penelitian kami menguraikan mengapa penggunaan CPU dan RAM tidak mungkin dianalisis menggunakan MN. Kami juga menunjukkan peluang dengan metode yang kami usulkan untuk menganalisis penggunaan sumber daya perangkat keras di masa mendatang.

Penelitian ini menggunakan *virtual machine* pada komoditas hardware, AMD Ryzen 3900x 24 CPU dan 80 GB RAM. Kami mengalokasikan *server host virtual* dengan 16 CPU dan 16 GB RAM. Setiap *node* NDN diberi 1 CPU dan 2048 MB RAM untuk memeriksa cara kerja *node* NDN untuk sumber daya perangkat keras

yang minimal. PNETLab dan Mini-NDN sama-sama memiliki alokasi resource *virtual host* yang sama, sehingga kita bisa membandingkan bagaimana respon kedua metode tersebut dalam menjalankan skenario. Dalam sistem PL yang kami usulkan, setiap *node* memiliki semua komponen NDN secara independen, *library*, komponen utama, NDN, *tools* dan alokasi antarmuka (bergantung pada koneksi yang diperlukan). Semua komponen NDN berjalan di Qemu, kemudian *node* NDN di sistem PL berfungsi sebagai sistem lengkap yang berjalan di VM, sementara sistem MN menggunakan fitur *Linux network name-space* untuk membedakan proses dan antarmuka untuk meniru *node* NDN.

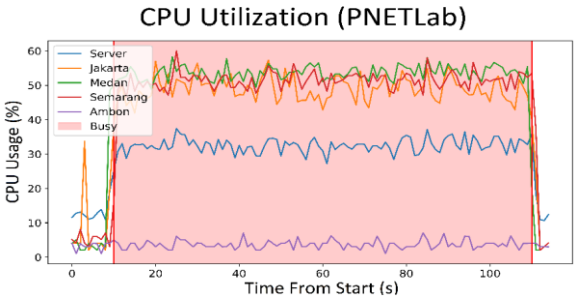


(a)

```
%CPU %MEM ARGS Fri Sep  2 13:58:53 WIB 2022
 0.6  0.0 /sbin/init
 0.6  0.1 nfd
 0.6  0.1 nfd
 0.6  0.1 nfd
 0.7  0.0 /usr/lib/snapd/snapd
 0.7  0.1 /usr/lib/xorg/Xorg
 0.7  0.2 /usr/bin/gnome-software
 0.7  0.0 /usr/bin/gedit
 1.6  0.0 /usr/lib/packagekit/packagekitd
 2.2  0.2 /usr/bin/gnome-shell
```

(b)

Gambar V.6 (a) Utilisasi CPU *Node* NDN & *server* host pada Mini-NDN; (b) CPU proses & penggunaan ram pada *node* NDN dan *server* host



(a)

```
%CPU %MEM ARGS Tue Sep 20 03:47:29 UTC 2022
 6.2  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 6.4  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 7.6  1.0 /usr/lib/jvm/default-java/bin/java
 7.8  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 8.2  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 8.7  1.3 /opt/qemu-2.12.0/bin/qemu-system-x86_64
 9.0  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64
11.6  1.4 /opt/qemu-2.12.0/bin/qemu-system-x86_64 -device
14.3  1.3 /opt/qemu-2.12.0/bin/qemu-system-x86_64 -device
15.8  1.3 /opt/qemu-2.12.0/bin/qemu-system-x86_64 -device
```

(b)

```
%CPU %MEM ARGS Sel Sep 20 10:40:42 WIB 2022
 0.0  0.0 bash
 0.0  0.1 ps
 0.0  0.0 cut
 0.0  0.0 tail
 0.1  0.3 /sbin/init
 0.1  2.3 /usr/bin/enlightenment
 0.2  1.9 /usr/lib/xorg/Xorg
 2.3  1.7 terminology
 5.0  0.7 nlsr
 7.9  3.7 /usr/local/bin/nfd
```

(c)

Gambar V.7 (a) Utilisasi CPU *Node* NDN & *server* host pada PNETLab; CPU proses & penggunaan ram (b) pada *server* host ; (c) pada *node* NDN

Pada pengamatan CPU Utilization, perbedaan metode emulasi dari PL dan MN sangat signifikan. Kontainerisasi proses sistem MN mengarahkan setiap proses untuk berjalan di OS *server* host. Konsekuensinya, kita tidak bisa melihat detail proses apa yang terjadi pada setiap *node* NDN menggunakan MN. Kami mencoba menangkap proses yang terjadi pada setiap *node* NDN (konsumen, produsen, dan *node* NDN perantara) dan *server* host untuk mendapatkan beban total pada setiap sumber daya perangkat keras. Ditemukan bahwa meskipun proses ditangkap pada *node* melalui jendela Linux X-Term di MN, daftar proses yang diperoleh identik untuk semua *node* dan *server* host, seperti yang ditunjukkan pada Gambar. V.6.(a) dan V.6.(b), meskipun setiap *node* memiliki aktivitas yang berbeda.

Sedangkan di PL, jika kita mengakses *node* NDN, kita dapat menangkap proses yang berbeda di antara *node* tersebut. Virtualisasi sistem lengkap Qemu menyediakan isolasi proses yang berjalan pada setiap *node* NDN, digambarkan pada Gambar V.7.(b) dan V.7.(c). Gambar V.7.(a) menunjukkan bahwa *node* Jakarta (produsen), Semarang, dan Medan (konsumen) memiliki beban CPU tertinggi. *Node* Ambon, yang hanya bertindak sebagai *node* perantara, memiliki beban CPU terendah. Oleh karena itu, dalam lingkungan emulasi PL, kita dapat melihat proses terperinci yang berjalan di setiap *node* NDN dan pemanfaatan proses tersebut untuk sumber daya perangkat keras.

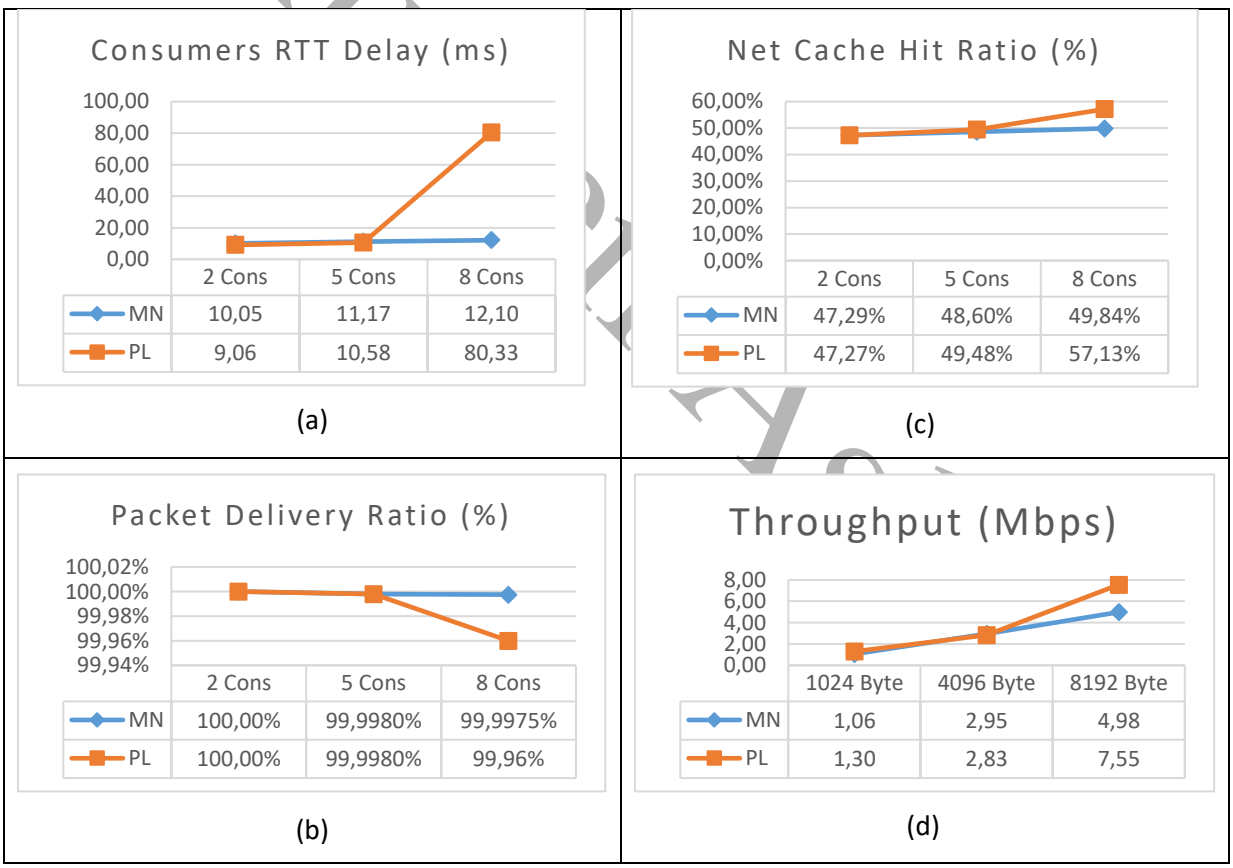
Pengukuran dalam PL untuk penggunaan RAM berfluktuasi dan tidak menunjukkan dampak langsung dari skenario yang kami buat. Itu karena RAM adalah satu-satunya buffer sementara untuk proses yang berjalan di *node* NDN. Namun, seperti yang terlihat pada tabel V.2, catatan data penggunaan RAM menunjukkan bahwa semua *node* NDN di PL beroperasi di bawah alokasi RAM 2048 MB. Ini berarti bahwa semua proses NDN tidak di-buffer untuk waktu yang lama untuk diproses oleh CPU.

Tabel V.2 PNETLab CPU, RAM, VRAM record

Time	% CPU used	RAM used (Bytes)	VRAM used (Bytes)
5:14:33 PM	5.376344	655519744	655519744
5:14:34 PM	2.040816	655519744	655519744

5:14:35 PM	4	655765504	655765504
5:14:36 PM	36.458333	720560128	720560128
5:14:37 PM	22.222222	754647040	754647040
5:14:38 PM	4.040404	754647040	754647040
5:14:39 PM	3.092784	754647040	754647040
5:14:40 PM	4.081633	754606080	754606080
5:14:41 PM	3.061224	754589696	754589696
5:14:42 PM	4.081633	755380224	755380224
5:14:43 PM	12.5	756912128	756912128
5:14:44 PM	5.050505	756871168	756871168

Pengamatan *Delay* Round Trip (RTT) dengan perubahan jumlah konsumen menunjukkan semakin banyak konsumen maka semakin besar pula *delay* komunikasi data pada jaringan NDN-IDN seperti pada Gambar V.8.(a). *Delay* RTT meningkat karena meningkatnya jumlah permintaan yang dilayani di jaringan. Setiap konsumen menghasilkan 100 interest/detik, yang berarti total 800 interest/detik untuk 8 (delapan) konsumen.



Gambar V.8 Experiment result: (a) Consummers RTT Delay; (b) Packet Delivery Ratio (PDR); (c) Network Cache Hit Tatio (nCHR), and (d) Throughput

Produser menanggapi permintaan pertama dari konsumen, sedangkan cache *node* perantara melayani permintaan berulang.

Seperti terlihat pada Gambar V.8.(a), delay yang signifikan terjadi ketika delapan konsumen melakukan permintaan secara bersamaan karena antrean untuk melayani semua permintaan. Emulasi NDN berbasis PL memiliki hasil yang mirip dengan emulasi MN ketika 2 (dua) konsumen dan 5 (lima) konsumen menghasilkan traffic minat. Namun demikian, emulasi NDN pada PL mengalami delay RTT yang tinggi pada 8 (delapan) pembangkitan trafik konsumen secara bersamaan. Delay RTT meningkat 8 (delapan) kali menjadi 80,33 ms.

Penurunan performa PL yang signifikan pada skenario 8 (delapan) konsumen disebabkan oleh terbatasnya alokasi sumber daya pada *node* NDN. Setiap NDN *node* mendapatkan alokasi 1 (satu) core CPU dan 2048 MB RAM yang dapat dikatakan sebagai resource minimal untuk perangkat jaringan. Di sisi lain, pembatasan sumber daya tidak dapat diterapkan pada MN, sehingga MN menggunakan semua sumber daya *server* host. Delay RTT akan meningkat secara signifikan ketika jaringan IDN-NDN diberi beban tinggi dengan 8 konsumen (menghasilkan total 800 *interest*/detik). Ini disebabkan oleh isolasi yang baik dari proses yang berjalan pada setiap *node* NDN di lingkungan PL.

Sebaliknya, ketika delay RTT meningkat secara signifikan, Net Network Cache Hit Ratio (nCHR) pada PL cenderung memberikan performa yang lebih baik pada 8 (delapan) skenario konsumen yang digambarkan pada Gambar V.8.(c). Pengamatan terhadap Network Cache Hit Ratio (nCHR) dengan perubahan jumlah konsumen menunjukkan bahwa peningkatan jumlah konsumen juga akan meningkatkan probabilitas hit cache. Ketentuan nCHR berarti permintaan yang muncul tidak diteruskan ke produsen untuk dilayani. *Node* perantara melayani permintaan di jaringan. Dengan nilai nCHR yang tinggi, beban produsen akan berkurang secara signifikan.

Namun, kenaikan nCHR tidak dapat mengompensasi total laju *interest* konsumen, sehingga Packet Delivery Ratio (PDR) turun menjadi 99,6% pada PL yang digambarkan pada Gambar V.8.(b). PDR adalah perbandingan rasio antara total paket yang dikirim ke semua paket yang ditransmisikan. Memberikan begitu banyak permintaan bunga/permintaan kedua dengan sumber daya perangkat keras yang minimal membuat antrean paket menjadi intens. Akibatnya, beberapa permintaan tidak dapat dipenuhi karena paket menunggu terlalu lama dalam antrean dan tidak dapat ditangani oleh sumber daya

NDN-IDN dengan peningkatan konsumen memberikan hasil yang lebih stabil untuk lingkungan emulasi MN karena tidak ada masalah batasan sumber daya. PL memberikan kinerja yang solid untuk skenario 2 (dua) konsumen dan 5 (lima) konsumen dimana kedua laju interest masing-masing adalah 200 dan 500 bunga/s.

Kemudian kami mengevaluasi kinerja throughput pada PL dan ML (lihat Gambar V.8.(d)). Tiga ukuran paket digunakan, 8.192, 4.096, dan 1.024 Bytes. Sebagai skenario paling stabil, kami menempati 2 (dua) konsumen dengan masing-masing 100 *interest*/s. Hasil menunjukkan PL dan ML memberikan kinerja yang dekat untuk ukuran paket 4k dan 1k Bytes, tetapi dalam 8k Bytes, PL memberikan throughput yang lebih tinggi daripada MN.

Peningkatan ini karena pengiriman konten berukuran besar membutuhkan lebih sedikit permintaan. Jadi, lebih banyak permintaan dapat dilayani dengan lebih sedikit permintaan dan ukuran paket yang lebih besar. Seperti diuraikan dalam Persamaan. (13), PS sebanding dengan throughput dalam kondisi non-blocking dan kecukupan sumber daya. Dengan demikian, lingkungan emulasi PL memberikan throughput yang lebih solid daripada MN untuk ukuran paket 8K Bytes.

Bab VI Kesimpulan dan Saran

Kesimpulan

- GSNR, dengan protokol perutean hibrid, memberikan kinerja yang lebih baik dan stabil daripada NLSR dan GEO.
- GSNR memiliki *overhead* pesan yang rendah tetapi tidak mengorbankan kinerja transmisi data.
- GSNR menunjukkan kinerja yang lebih stabil untuk parameter throughput dan delay, dengan perubahan ukuran area dan jumlah *node*. Performa GSNR (*overhead*, path-stretch, delay, throughput) rata-rata hanya mengalami perubahan (naik atau turun) 19,83% dibandingkan NLSR 116,46% dan GEO 50,91%.
- GSNR memberikan hasil yang baik dengan kompleksitas yang juga cukup sederhana dengan nilai Big O adalah $O(n)$.
- Telah dibangun juga model perbandingna efisiensi antara perutean GR dan LS.
- Untuk perbandingan keseluruhan, hasilnya menunjukkan bahwa protokol perutean berbasis LS mengungguli protokol perutean berbasis GR. Perutean LS memiliki kinerja yang lebih baik dengan efisiensi rata-rata 23,8%.
- Multi koneksi dapat mencapai peningkatan efisiensi untuk setiap sesi, lebih sedikit hop/jarak, ukuran header paket yang rasional, dan ukuran paket yang lebih besar untuk membawa muatan data.
- Selanjutnya juga diusulkan sebuah metode sebagai katalis untuk mempercepat penelitian NDN menuju implementasi dunia nyata. Metode ini menempati lingkungan virtualisasi berbasis UNetLab untuk observasi ekstensif bagi peneliti.
- Sebagai bukti yang menguatkan metode ini, kami membuat kasus uji di PNETLab dibandingkan dengan Mini-NDN sebagai mekanisme emulasi NDN yang banyak digunakan. Hasilnya menunjukkan metode PL dapat memberikan pengukuran yang luas dari *node* NDN, sedangkan MN gagal.

- Namun, PL dan MN memberikan hasil yang sama pada pengukuran kinerja jaringan IDN-NDN untuk 100 hingga 500 *interest*/s, 7,59% pada delay RTT, dan 0,89% pada nCHR.
- PL maupun MN dapat digunakan untuk menganalisis kinerja NDN. Selain itu, hanya PL yang dapat memberikan data detail tentang proses penggunaan perangkat keras untuk setiap *node*, yang sangat penting untuk implementasi NDN.
- Dari pengukuran yang sudah dilakukan pada PL dapat disimpulkan bahwa NDN dapat diterapkan pada NDN-IoT dengan resource minimal (1 CPU dan 2 GB RAM)

Saran

Penelitian lanjutan masih sangat terbuka untuk mengembangkan teknik perutean yang *scalable* pada Named Data Network. Perutean yang dapat beradaptasi untuk lingkungan jaringan NDN yang masif dalam skala ukuran yang luas. Terkait dengan pengembangan algoritma GSNR dapat dilanjutkan hingga tahap hilirisasi dengan kerangka kerja yang telah diusulkan yaitu PL. Sehingga pengembangan implementasi *Named Data Network* dapat dilakukan pula untuk membawa NDN ke arah riil internet masa depan.

DAFTAR PUSTAKA

Aboud, A., and Touati, H. (2016): Geographic interest forwarding in NDN-based wireless sensor networks, *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, IEEE, 1–8. https://doi.org/10.1109/AICCSA.2016.7945683

Afsar, M. M., and Tayarani-N, M.-H. (2014): Clustering in sensor networks: A literature survey, *Journal of Network and Computer Applications*, **46**, 198–226. https://doi.org/10.1016/J.JNCA.2014.09.005

Amadeo, M., Campolo, C., Ruggeri, G., Lia, G., and Molinaro, A. (2020): Caching transient contents in vehicular named data networking: A performance analysis, *Sensors (Switzerland)*, **20**(7), 1–17. https://doi.org/10.3390/s20071985

Amadeo, M., Ruggeri, G., Campolo, C., and Molinaro, A. (2019): IoT services allocation at the edge via named data networking: From optimal bounds to practical design, *IEEE Transactions on Network and Service Management*, **16**(2), 661–674. https://doi.org/10.1109/TNSM.2019.2900274

Ariefianto, W. T., and Syambas, N. R. (2017): Routing in NDN network: A survey and future perspectives, *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, IEEE, 1–6. https://doi.org/10.1109/TSSA.2017.8272942

Berger, L. (2008): *RFC 5250 - The OSPF Opaque LSA Option*, retrieved September 20, 2017from internet: https://tools.ietf.org/pdf/rfc5250.pdf.

BGP Reports. (n.d.): , retrieved March 12, 2019, from internet: http://bgp.potaroo.net./.

Boguñá, M., Papadopoulos, F., and Krioukov, D. (2010): Sustaining the Internet with hyperbolic mapping, *Nature Communications*, **1**(6), 1–8. https://doi.org/10.1038/ncomms1063

Breslau, L., Pei Cao, Li Fan, Phillips, G., and Shenker, S. (1999): Web caching and Zipf-like distributions: evidence and implications, *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, IEEE, 126–134 vol.1. https://doi.org/10.1109/INFCOM.1999.749260

Buchroithner, M. F., and Pfahlbusch, R. (2017): Geodetic grids in authoritative maps – new findings about the origin of the UTM Grid, *Cartography and Geographic Information Science*, **44**(3), 186–200. https://doi.org/10.1080/15230406.2015.1128851

Budiana, M. S., Nadra, M. M., Hapsari, T. R., Mayasari, R., and Syambas, N. R. (2021): Impact of the Content Store Scaling toward the LRU and FIFO Cache Replacements on NDN using Mini-NDN, *Proceeding of 15th International Conference on Telecommunication Systems, Services, and Applications, TSSA 2021*. https://doi.org/10.1109/TSSA52866.2021.9768288

CCN web. (n.d.): , retrieved from internet: https://wiki.fd.io/view/Cicn.

Charpinel, S., Santos, C. A. S., Vieira, A. B., Villaca, R., and Martinello, M. (2016): SDCCN: A Novel Software Defined Content-Centric Networking Approach, *2016 IEEE 30th International Conference on Advanced*

Information Networking and Applications (AINA), IEEE, 87–94. https://doi.org/10.1109/AINA.2016.86

Detti, A, Pomposini, M., Blefari-Melazzi, N., and Salsano, S. (2012): Supporting the Web with an information centric network that routes by name, *Computer Networks*, **56**, 3705–3722. https://doi.org/10.1016/j.comnet.2012.08.006

Detti, Andrea, Blefari Melazzi, N., Salsano, S., and Pomposini, M. (2011): CONET: A Content Centric Inter-Networking Architecture, *Proceedings of the ACM SIGCOMM workshop on Information-centric networking - ICN '11*, ACM Press, New York, New York, USA, 50. https://doi.org/10.1145/2018584.2018598

EVE-NG Site. (n.d.): , retrieved July 30, 2022, from internet: https://www.eve-ng.net/.

Friyanto, A., Ariefianto, W. T., and Syambas, N. R. (2019): Analysis Operation NLSR With Ubuntu as NDN Router, *2019 IEEE 5th International Conference on Wireless and Telematics (ICWT)*, IEEE, 1–4. https://doi.org/10.1109/ICWT47785.2019.8978267

Gameiro, L., Senna, C., and Luís, M. (2022): Insights from the Experimentation of Named Data Networks in Mobile Wireless Environments, *Future Internet*, **14**(7), 1–18. https://doi.org/10.3390/fi14070196

Gao, J. (2009): Geometric Routing in Wireless Sensor Networks, 113–157 *in Guide to Wireless Sensor Networks*, Springer. https://doi.org/10.1007/978-1-84882-218-4_5

GitHub - GNS3/dynamips: Dynamips development. (n.d.): , retrieved July 30, 2022, from internet: https://github.com/GNS3/dynamips.

Guo, X., Yang, S., Cao, L., Wang, J., and Jiang, Y. (2021): A new solution based on optimal link-state routing for named data MANET, *China Communications*, **18**(4), 213–229. https://doi.org/10.23919/JCC.2021.04.016

Home - Docker. (n.d.): , retrieved July 30, 2022, from internet: https://www.docker.com/.

Hoque, A. K. M. M., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L., and Wang, L. (2013): NLSR: Named-data Link State, *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking - ICN '13*, ACM Press, New York, New York, USA, 15. https://doi.org/10.1145/2491224.2491231

Interest Packet — NDN Packet Format Specification 0.2-2 documentation. (n.d.): , retrieved November 16, 2017, from internet: https://named-data.net/doc/ndn-tlv/interest.html.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009): Networking named content, *Proceedings of the 5th international conference on Emerging networking experiments and technologies - CoNEXT '09*, ACM Press, New York, New York, USA, 1. https://doi.org/10.1145/1658939.1658941

Johnson, D., Hu, Y., and Maltz, U. D. (2007): *RFC 4728 - The Dynamic Source Routing Protocol …DSR— for Mobile Ad Hoc Networks for IPv4*, retrieved April 28, 2019from internet: https://tools.ietf.org/pdf/rfc4728.pdf.

Kaimin Wei, Xiao Liang, and Ke Xu (2014): A Survey of Social-Aware Routing Protocols in Delay Tolerant Networks: Applications, Taxonomy and Design-Related Issues, *IEEE Communications Surveys & Tutorials*, **16**(1), 556–578.

https://doi.org/10.1109/SURV.2013.042313.00103
Karp, B., and Kung, H. T. (2000): GPSR: Greedy Perimeter Stateless Routing for Wireless Network, *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*, ACM Press, New York, New York, USA, 243–254. https://doi.org/10.1145/345910.345953
Kleinberg, R. (2007): Geographic Routing Using Hyperbolic Space, *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, IEEE, 1902–1909. https://doi.org/10.1109/INFCOM.2007.221
Kleinrock, L., and Kamoun, F. (1977): Hierarchical routing for large networks Performance evaluation and optimization, *Computer Networks (1976)*, **1**(3), 155–174. https://doi.org/10.1016/0376-5075(77)90002-2
Krioukov, D., Papadopoulos, F., Boguñá, M., and Vahdat, A. (2009): Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces, *ACM SIGMETRICS Performance Evaluation Review*, **37**(2), 15. https://doi.org/10.1145/1639562.1639568
Krishnamurthy, B., and Wang, J. (2000): On network-aware clustering of Web clients, *ACM SIGCOMM Computer Communication Review*, **30**(4), 97–110. https://doi.org/10.1145/347057.347412
Kuhn, F., Wattenhofer, R., Zhang, Y., and Zollinger, A. (2003): Geometric ad-hoc routing: Of Thory an Practice, *Proceedings of the twenty-second annual symposium on Principles of distributed computing - PODC '03*, ACM Press, New York, New York, USA, 63–72. https://doi.org/10.1145/872035.872044
Lehman, V., Chowdhury, M., and Gordon, N. (2017): NLSR Developer ' s Guide, 1–20.
Lehman, V., Gawande, A., Zhang, B., Zhang, L., Aldecoa, R., Krioukov, D., and Wang, L. (2016): An experimental investigation of hyperbolic routing with a smart forwarding plane in NDN, *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, IEEE, 1–10. https://doi.org/10.1109/IWQoS.2016.7590394
Lehman, V., and Wang, L. (2016): *A Secure Link State Routing Protocol for NDN*, retrieved from internet: https://named-data.net/wp-content/uploads/2016/01/ndn-0037-1-nlsr.pdf.
Liang Qin, and Kunz, T. (2006): Mobility Metrics to Enable Adaptive Routing in MANET, *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2006. (WiMob'2006)*, IEEE, 1–8. https://doi.org/10.1109/WIMOB.2006.1696350
Mastorakis, S., Afanasyev, A., and Zhang, L. (2017): On the Evolution of ndnSIM, *ACM SIGCOMM Computer Communication Review*, **47**(3), 19–33. https://doi.org/10.1145/3138808.3138812
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008): OpenFlow: Enabling Innovation in Campus Networks, *ACM SIGCOMM Computer Communication Review*, **38**(2), 69. https://doi.org/10.1145/1355734.1355746
Meyer, D. (2007): *RFC 4984 - Report from the IAB Workshop on Routing and Addressing*, retrieved April 5, 2018from internet: https://tools.ietf.org/pdf/rfc4984.pdf, 1–39.
Mosko, M., Solis, I., and Wood, C. (2019): *draft-irtf-icnrg-ccnxmessages-09* -

CCNx Messages in TLV Format, retrieved February 26, 2019from internet: http://datatracker.ietf.org/drafts/current/.

Moy, J. (1998): *RFC 2328 - OSPF Version 2*, retrieved September 20, 2017from internet: https://tools.ietf.org/pdf/rfc2328.pdf.

ndn-traffic-generator_modified. (n.d.): , retrieved July 31, 2022, from internet: https://github.com/aderama2711/ndn-traffic-generator_modified.

Ofelia Project. (n.d.): , retrieved from internet: http://www.fp7-ofelia.eu/.

ONF Web. (n.d.): , retrieved from internet: https://www.opennetworking.org/.

Ooka, A., Ata, S., Koide, T., and Shimonishi, H. (2013): OpenFlow-based Content-Centric Networking Architecture and Router Implementation, *Future Network and Mobile Summit (FutureNetworkSummit), 2013*, retrieved September 24, 2017from internet: http://www-mura.ist.osaka-u.ac.jp/achievements/web2013/paper/a-ooka13funems-OFbasedCCN.pdf, 1–10.

Opendaylight web. (n.d.): , retrieved from internet: https://www.opendaylight.org/.

Perkins, C. E., and Royer, E. M. (1999): Ad-hoc on-demand distance vector routing, *Proceedings - WMCSA'99: 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 90–100. https://doi.org/10.1109/MCSA.1999.749281

PNETLab : Lab is Simple Site. (n.d.): , retrieved July 30, 2022, from internet: https://pnetlab.com/pages/main.

RFC 3040 - Internet Web Replication and Caching Taxonomy (2001): , retrieved October 7, 2017from internet: https://tools.ietf.org/pdf/rfc3040.pdf.

Salsano, S., Blefari-Melazzi, N., Detti, A., Morabito, G., and Veltri, L. (2013): Information centric networking over SDN and OpenFlow: Architectural aspects and experiments on the OFELIA testbed, *Computer Networks*, **57**(16), 3207–3221. https://doi.org/10.1016/j.comnet.2013.07.031

Saxena, D., and Raychoudhury, V. (2019): Design and Verification of an NDN-Based Safety-Critical Application: A Case Study with Smart Healthcare, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **49**(5), 991–1005. https://doi.org/10.1109/TSMC.2017.2723843

Saxena, D., and Roorkee, I. I. T. (2016): Named Data Networking: A Survey, *Computer Science Review, Elsevier*, **19**, 15--55. https://doi.org/10.1016/j.cosrev.2016.01.001

Torres, J. V., Ferraz, H. G., Carlos, O., and Duarte, M. B. (2012): *Controller-based Routing Scheme for Named Data Network*, retrieved October 4, 2017from internet: https://www.gta.ufrj.br/ftp/gta/TechReports/TFD12.pdf, 1–6.

Total number of Websites - Internet Live Stats. (n.d.): , retrieved March 12, 2019, from internet: http://www.internetlivestats.com/total-number-of-websites/.

Tubaishat, M., Yin, J., Panja, B., and Madria, S. (2004): A secure hierarchical model for sensor network, *ACM SIGMOD Record*, **33**(1), 7. https://doi.org/10.1145/974121.974123

Wang, L., and Hoque, A. K. M. M. (2012): OSPFN : An OSPF Based Routing Protocol for Named Data Networking, 1–15.

Wang, L., Lehman, V., Mahmudul Hoque, A. K. M., Zhang, B., Yu, Y., and Zhang, L. (2018): A Secure Link State Routing Protocol for NDN, *IEEE*

Access, **6**, 10470–10482. https://doi.org/10.1109/ACCESS.2017.2789330

Wibowo, T. A., Syambas, N. R., and Hendrawan (2018): Overhead of Named Data Networking Routing Protocol, *2018 12th International Conference on Telecommunication Systems Services and Applications (TSSA)*, IEEE, 1–5.

Wibowo, T. A., Syambas, N. R., and Hendrawan (2019): Named Data Network (NDN) Scalability Problem, *IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*.

Wibowo, T. A., Syambas, N. R., and Hendrawan, H. (2020): The routing protocol efficiency of named data network, *Proceeding of 14th International Conference on Telecommunication Systems, Services, and Applications, TSSA 2020*, Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/TSSA51342.2020.9310886

WorldWideWebSize.com | The size of the World Wide Web (The Internet). (n.d.): , retrieved March 12, 2019, from internet: http://www.worldwidewebsize.com/.

Xu, S., Blackmore, K. L., and Jones, H. M. (2007): An Analysis Framework for Mobility Metrics in Mobile Ad Hoc Networks, *EURASIP Journal on Wireless Communications and Networking*. https://doi.org/10.1155/2007/19249

Yi, C., Abraham, J., Afanasyev, A., Wang, L., Zhang, B., and Zhang, L. (2014): On the role of routing in named data networking, *Proceedings of the 1st international conference on Information-centric networking - INC '14*, ACM Press, New York, New York, USA, 27–36. https://doi.org/10.1145/2660129.2660140

Yi, C., Afanasyev, A., Moiseenko, I., Wang, L., Zhang, B., and Zhang, L. (2013): A case for stateful forwarding plane, *Computer Communications*, **36**(7), 779–791. https://doi.org/10.1016/j.comcom.2013.01.005

Yovita, L. V., Syambas, N. R., Edward, I. J. M., and Kamiyama, N. (2020): Performance analysis of cache based on popularity and class in named data network, *Future Internet*, **12**(12), 1–22. https://doi.org/10.3390/fi12120227

Yuan, H., Song, T., and Crowley, P. (2012): Scalable NDN Forwarding: Concepts, Issues and Principles, *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, IEEE, 1–9. https://doi.org/10.1109/ICCCN.2012.6289305

Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Claffy, K., Crowley, P., Papadopoulos, C., Wang, L., and Zhang, B. (2014): Named data networking, *ACM SIGCOMM Computer Communication Review*, **44**(3), 66–73. https://doi.org/10.1145/2656877.2656887

Zhu, Z., and Afanasyev, A. (2013): Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking, *2013 21st IEEE International Conference on Network Protocols (ICNP)*, IEEE, 1–10. https://doi.org/10.1109/ICNP.2013.6733578

Zipf, G. K. (1929): Relative Frequency as a Determinant of Phonetic Change, *Harvard Studies in Classical Philology*, **40**, 1. https://doi.org/10.2307/310585