DOI: 10.22146/jnteti.v13i2.10185

Analisis Kinerja Sistem Pemantauan Berbasis *Streaming Telemetry* gNMI dengan Simulasi Jaringan Containerlab

Fierda Kurniacahya Ariefputra[1], Eueung Mulyana[1]

[1] Departemen Teknik Elektro, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, Jawa Barat 40132, Indonesia

[Diserahkan: 24 Oktober 2023, Direvisi: 27 Desember 2023, Diterima: 21 Maret 2024]
Penulis Korespondensi: Fierda Kurniacahya Ariefputra (email: 23221081@mahasiswa.itb.ac.id)

INTISARI — Pesatnya pertumbuhan internet berdampak pada perkembangan layanan digital. Lonjakan permintaan ini telah menciptakan peluang bagi pelaku industri layanan digital. Meskipun memiliki dampak positif, pertumbuhan internet juga menimbulkan tantangan teknis. Dalam konteks mengatasi pertumbuhan lalu lintas data, pemantauan sumber daya (*resource*) menjadi salah satu faktor yang krusial. Salah satu cara terbaru untuk memantau sumber daya tersebut adalah dengan menggunakan sistem *streaming telemetry* Google Remote Procedure Call (gRPC) Network Management Interface (gNMI). Meski tampak lebih superior dari protokol yang telah ada, studi lebih lanjut dalam pengimplementasian sistem *streaming telemetry* perlu dikaji. Makalah ini difokuskan untuk menginvestigasi *trade-off* dan kinerja *streaming telemetry* gNMI. Perancangan dan simulasi dilakukan menggunakan *networking lab tools* berbasis Docker, yaitu containerlab. Integrasi sistem pemantauan dan topologi jaringan juga dilakukan dalam simulasi berbasis Docker tersebut. Hasil pengamatan dari tiap protokol menunjukkan bahwa aktivitas mengambil metrik oleh sistem pemantauan tidak berpengaruh signifikan terhadap kondisi jaringan. Hal tersebut ditunjukkan oleh rata-rata *latency* jaringan yang bervariasi rendah dan *throughput* yang hampir seragam kecuali pada kondisi *lossy* dan *congestion*. Pengamatan pada simulasi juga menunjukkan sistem pemantauan gNMI menggunakan sumber daya *input*/*output* (I/O) lebih intensif dibandingkan protokol lainnya. Penelitian ini juga membahas tentang integrasi *streaming telemetry* gNMI dan *log monitoring*, yang menunjukkan kenaikan penggunaan memori sebesar 70 MB dan sumber daya *Disk I/O* yang meningkat sebesar 33%. Dalam penelitian ini juga ditemukan indikasi peningkatan penggunaan CPU oleh sistem pemantauan gNMI sebesar 50% dari data modus yang dicatat dalam pengamatan.

KATA KUNCI — Pemantauan Jaringan, *Streaming Telemetry*, gRPC Network Management Interface, Jaringan Pusat Data, Simulasi.

I. PENDAHULUAN

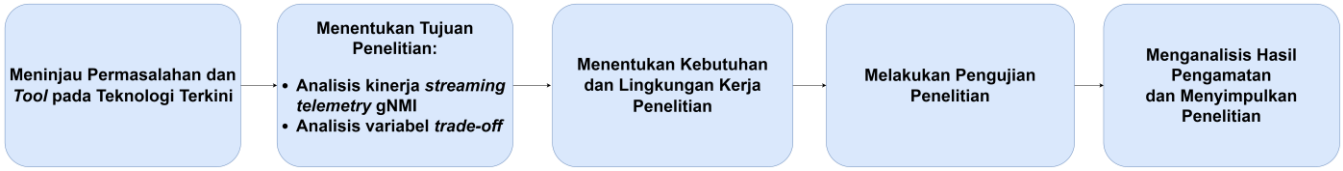
Pertumbuhan penggunaan internet telah memberikan dampak yang sangat besar di berbagai sektor, yang secara signifikan memengaruhi kualitas hidup manusia [1], [2]. Jaringan pertukaran data yang saling terhubung ini telah merevolusi berbagai bidang, antara lain komunikasi, pendidikan, hiburan, dan perdagangan. Internet telah menjadi bagian integral dari kehidupan sehari-hari, memberdayakan individu dengan akses mudah ke informasi, dan menghubungkan orang-orang di seluruh dunia.

Pesatnya pertumbuhan internet tersebut juga berdampak besar pada perkembangan pelaku usaha yang bergerak di bidang layanan *hosting* digital dan layanan internet. Karena tingkat penetrasi internet terus meningkat, makin banyak individu dan pelaku bisnis yang mencari layanan *hosting* dan internet yang andal untuk membangun sistem *online*. Tidak hanya perusahaan besar dan multinasional yang mengambil kesempatan ini, tetapi juga perseroan skala kecil dan menengah dengan cakupan nasional, yang juga tumbuh signifikan [3]. Lonjakan permintaan ini telah menciptakan peluang yang signifikan bagi pelaku bisnis dan industri di sektor layanan digital dan layanan internet. Hal tersebut memungkinkan para pelaku bisnis dan industri untuk berkembang dan memperluas operasi.

Meskipun memiliki banyak dampak positif di berbagai bidang, pertumbuhan internet yang pesat juga menimbulkan tantangan dalam hal teknis, khususnya dalam pengelolaan lalu lintas jaringan komunikasi dan sumber daya komputasi. Hal tersebut diakibatkan oleh makin banyaknya perangkat yang terhubung ke internet, sehingga permintaan *bandwidth* jaringan dan kebutuhan penyimpanan data meningkat secara eksponensial. Lonjakan konektivitas ini dapat menyebabkan masalah, seperti kepadatan jaringan, penurunan kecepatan transfer data, dan terbatasnya sumber daya untuk layanan dan aplikasi pada layanan *hosting* [4].

Salah satu cara terbaru untuk memantau sumber daya infrastruktur teknologi informasi adalah dengan menggunakan sistem berbasis *streaming telemetry*. Sistem ini melakukan pengumpulan dan analisis data secara *real-time* dari berbagai perangkat jaringan dan sumber daya komputasi. Sistem pemantauan ini memberikan wawasan berkelanjutan tentang kinerja, pemanfaatan, dan perilaku infrastruktur teknologi informasi. *Streaming telemetry* memungkinkan operator memantau pola lalu lintas jaringan, mengidentifikasi potensi hambatan aliran data, dan mengambil tindakan proaktif, terutama untuk mengoptimalkan dan memastikan aliran data yang efisien (lebih rendah *overhead*).

Dalam pendekatan pemantauan jaringan (*network monitoring*) tradisional, metode pemantauan sumber daya dengan Simple Network Management Protocol (SNMP) digunakan untuk mengumpulkan data secara berkala [5], [6]. Metode ini digunakan untuk memantau metriks perangkat jaringan. Dalam pendekatan terbaru seperti *streaming telemetry*, pemantauan sistem dan jaringan memanfaatkan



Gambar 1. Metodologi penelitian.

protokol seperti Google Remote Procedure Call (gRPC) Network Management Interface (gNMI), yang menyediakan *streaming* metrik, *log*, dan *data trace* berkelanjutan dari berbagai sumber di dalam pusat data dan server [7], [8]. Pendekatan ini dapat diintegrasikan dengan aplikasi *microservice resource monitoring* yang terkemas dalam lingkungan virtualisasi serta dapat diintegrasikan dengan siklus *continuous integration and continuous delivery/continuous deployment* (CI/CD).

Meski tampak lebih canggih dan efisien dari protokol yang telah ada, studi lebih lanjut dalam strategi pengimplementasian sistem *streaming telemetry* perlu dilakukan, terutama terkait kinerja sistem *streaming telemetry* ketika dikonfigurasi untuk bekerja mendekati kondisi *real-time* dan dikondisikan dalam lingkungan sistem terdistribusi (lingkungan komputasi awan). Penelitian mengenai penggunaan dan implementasi tiap teknologi pemantauan dan telemetri jaringan telah dilakukan oleh beberapa peneliti maupun praktisi industri berpengalaman. Dengan meninjau beberapa penelitian yang telah dilakukan sebelumnya, fokus konteks penelitian dapat ditentukan, kesenjangan penelitian terdahulu diidentifikasi, dan desain penelitian ditetapkan.

Penelitian tentang pengimplementasian sistem telemetri berbasis gNMI pada jaringan serat optik dan *data center* telah dilakukan oleh beberapa peneliti. Referensi [9], [10] membahas implementasi dua komponen teknologi untuk manajemen *transceiver* dan pemantauan telemetri pada jaringan serat optik. Pemantauan telemetri dilakukan menggunakan protokol gNMI dan model perangkat OpenConfig. Penelitian tersebut mendemonstrasikan kelayakan pendekatan teknologi ini dengan mendapatkan pengukuran telemetri *real-time* serta menganalisis skalabilitas solusi yang diusulkan terhadap jumlah *node transponder*. Beberapa penelitian lain juga memaparkan penerapan *streaming telemetry* yang diterapkan pada jaringan serat optik dan *data center* yang kompleks dan digunakan untuk mendeteksi pola lalu lintas yang berfungsi mendeteksi perubahan *state* dan *soft-failure* dalam jaringan serat optik [11]–[17].

Referensi [18] membandingkan kinerja pemantauan jaringan berbasis SNMP dengan *streaming telemetry* jaringan dengan kerangka gRPC dan Google Protocol Buffers (GPB). Hasil pengukuran menunjukkan bahwa pengambilan statistik *ifTable* yang dikodekan dengan Compact-GPB kira-kira dua setengah kali lebih efisien *bandwidth*-nya daripada *polling* SNMP dengan GetBulk. *Delay* jaringan *round-trip* terukur, termasuk paketisasi, serialisasi, dan pemrosesan aliran telemetri kira-kira empat kali lebih rendah daripada SNMP. Selain itu, pengukuran menunjukkan bahwa pemantauan jaringan dengan *streaming telemetry* lebih efisien dibandingkan *polling* SNMP dalam hal penggunaan CPU.

Sementara itu, telah dilakukan juga penelitian dengan membandingkan dua solusi pemantauan jaringan, yaitu berbasis SNMP serta berbasis Protocol Buffers dan gRPC [19]. Hasil pengukuran menunjukkan bahwa saat mengambil informasi tentang objek yang memiliki nilai 1 byte, SNMP memiliki kinerja lebih baik. Saat objek metrik dengan nilai lebih besar diambil, SNMP bekerja paling baik hingga 26 objek telemetri yang diambil per pesan. Di atas titik ini, kombinasi Protocol Bufffers dan gRPC bekerja lebih baik, yang menghasilkan lebih sedikit data (dalam *byte*) yang dikirim untuk sejumlah objek tertentu. Tidak ada dampak pada penggunaan memori dan CPU di *router* yang ditunjukkan.

Dalam makalah ini, penelitian difokuskan untuk mengetahui kinerja *streaming telemetry* dalam melakukan pemantauan kondisi jaringan dan sumber daya di simulasi lingkungan virtualisasi *data center*. Kinerja yang dimaksud antara lain pengaruh *telemetry stack* terhadap penggunaan sumber daya di perangkat *host*, pengaruh konfigurasi durasi interval, dan jumlah metrik yang dilanggan oleh *streaming telemetry*. Penelitian ini juga mempelajari perilaku lalu lintas jaringan terhadap parameter metriks yang sedang dipantau. Penelitian ini dilakukan dalam lingkungan simulasi virtualisasi berbasis Docker dan skenario penelitian ini adalah melakukan pemantauan sumber daya di perangkat jaringan yang berjalan pada *data center* dengan protokol komunikasi Border Gateway Protocol Ethernet Virtual Private Network (BGP EVPN VXLAN) [20].

II. METODOLOGI

Penelitian ini berfokus untuk melengkapi celah informasi, yaitu informasi kinerja dan variabel *trade-off* dari *streaming telemetry* dari publikasi, bahasan artikel, dan makalah penelitian yang telah dipublikasi dan dilakukan. Variabel *trade-off* yang dimaksud dalam penelitian ini adalah durasi *scraping interval* dan jumlah objek yang dilanggan oleh *streaming telemetry* gNMI. Metode penelitian ini ditunjukkan pada Gambar 1, yang menjelaskan langkah-langkah dan alur kerja yang dilakukan pada penelitian ini. Pengamatan dilakukan dengan cara mengamati nilai parameter secara manual, lalu mengambil data secara modus dari nilai yang ditampilkan oleh *tool* Glances selama durasi pengamatan berlangsung.

A. gRPC NETWORK MANAGEMENT INTERFACE (gNMI)

gNMI merupakan protokol ekstensi dari gRPC yang dikembangkan oleh OpenConfig dan distandarisasi oleh Internet Engineering Task Force (IETF) untuk menyediakan standar cara bagi perangkat jaringan untuk dikelola dan dipantau menggunakan mekanisme Remote Procedure Call (RPC) [21]. Cara kerja protokol ini dimulai dengan klien (yang merupakan suatu sistem pemantauan atau aplikasi) membuat koneksi gRPC dengan server gNMI yang berjalan di perangkat jaringan. Koneksi ini dapat dibuat melalui berbagai protokol *transport*, seperti Transmission Control Protocol (TCP), Transport Layer Security (TLS), atau gRPC melalui HTTP/2. Klien mengautentikasi dirinya sendiri ke server menggunakan kredensial atau sertifikat untuk memastikan komunikasi yang aman. Server memverifikasi identitas klien dan mengesahkan operasi yang diminta berdasarkan hak istimewa klien [22].

Klien mengirimkan permintaan gNMI ke server untuk melakukan operasi seperti mengambil data konfigurasi,

TABEL I
TOOLS DAN VERSI DALAM PERANCANGAN SISTEM UNTUK PENELITIAN

Nama *Tools*	Versi
Virtualbox	6.1.40 r154048
Linux Mint OS	20.2
Docker	20.10.21
Containerlab	0.42.0
Nokia SR Linux (*container* Docker)	23.2.2
Telegraf (*container* Docker)	1.27.0
influxDB (*container* Docker)	2.7.1
sFlow-RT (*container* Docker)	3.0-1676
Prometheus (*comtainer* Docker)	2.37.8
Grafana (*container* Docker)	9.5.2
gNMIc (*container* Docker)	2.3.0

mengubah pengaturan, berlangganan pembaruan status, atau menanyakan informasi operasional. Permintaan ditentukan menggunakan API Protocol Buffers gNMI, yang menentukan tindakan dan parameter yang diinginkan. Server menerima permintaan gNMI, memvalidasinya, dan memprosesnya sesuai dengan permintaan tersebut. gNMI dapat berinteraksi dengan sistem operasi jaringan yang mendasari atau *driver* perangkat jaringan untuk melakukan tindakan yang diminta, seperti membaca atau memodifikasi data konfigurasi. Setelah memproses permintaan, server mengirimkan kembali respons gNMI ke klien. Respons tersebut berisi informasi yang diminta atau status keberhasilan/kegagalan operasi. Respons juga dikodekan menggunakan Protocol Buffers.

Selain interaksi permintaan-respons, gNMI mendukung mekanisme berbasis langganan, yaitu klien dapat berlangganan untuk menerima informasi pembaruan atau pemberitahuan asinkron tentang perubahan atau peristiwa data tertentu. Server mendorong pembaruan ini ke klien seketika saat terjadi, memungkinkan pemantauan *real-time* dan manajemen berbasis *event*.

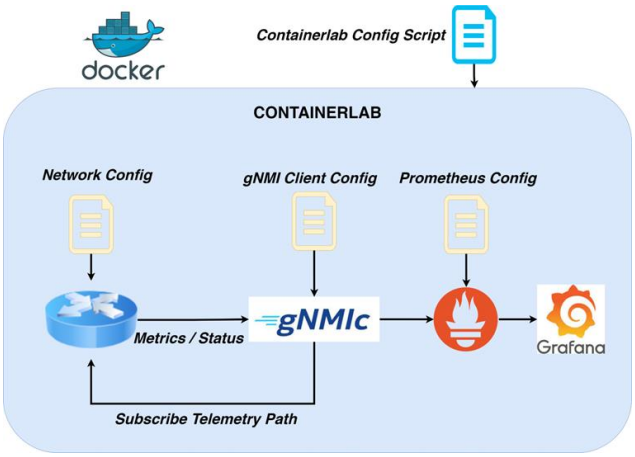
B. SPESIFIKASI TOOLS DAN PERANGKAT LUNAK SISTEM PEMANTAUAN

Perancangan sistem dalam penelitian ini menggunakan beberapa perangkat lunak atau *networking tools* berbasis *open source*. Penggunaan *open source tools* dalam penelitian ini bertujuan menjadikan penelitian ini transparan, sehingga peneliti atau praktisi industri dapat mereplika skenario dalam penelitian ini. Hal ini juga memungkinkan validasi dan verifikasi hasil penelitian serta dapat memastikan bahwa sistem yang dirancang berfungsi sebagaimana mestinya. Selain itu, sifat fleksibilitas dan mudah disesuaikan yang disediakan oleh *open source tool* menjadi pertimbangan dalam penelitian ini. Alasan tersebut menjadi dasar pemilihan simulasi jaringan containerlab untuk digunakan dalam penelitian ini.

Tools dan perangkat lunak yang digunakan dalam penelitian ditunjukkan pada Tabel I. Tabel tersebut menunjukkan juga versi yang digunakan dalam penelitian. Spesifikasi perangkat keras komputer yang digunakan adalah CPU Intel Core i7 2,6 GHz dengan enam inti dan *random access memory* (RAM) sebesar 16 GB. *Virtual machine* Linux Mint dikonfigurasi dengan prosesor empat inti dan RAM 8 GB.

C. KONFIGURASI NETWORK MONITORING STACK

Penelitian ini dilakukan untuk menguji kinerja dan mengetahui tingkat penggunaan sistem pemantauan dari beberapa protokol yang dipaparkan, terutama dalam kinerjanya memantau metriks sumber daya dan jaringan di simulasi *data center*. Protokol yang dimaksud adalah SNMP, sFlow, dan



Gambar 2. Arsitektur subsistem gNMI dalam penelitian.

gNMI. Protokol dalam bahasan ini dibagi menjadi tiga subsistem pemantauan, yang terdiri atas beberapa *tools* (*monitoring stack*) untuk mengambil, mengolah, dan menampilkan data metrik dari sistem yang dipantau.

Masing-masing subsistem pemantauan memiliki komponen *monitoring tools* dengan fungsi yang hampir sama. Seluruh *tools* yang digunakan dijalankan dalam lingkungan *container* Docker tunggal terpisah, kecuali pada *image* Docker sFlow-RT. Komunikasi antar *tools* dan jaringan simulasi *data center* dikelola oleh containerlab. Masing-masing subsistem menggunakan cara yang berbeda untuk memperoleh metrik atau status dari perangkat jaringan.

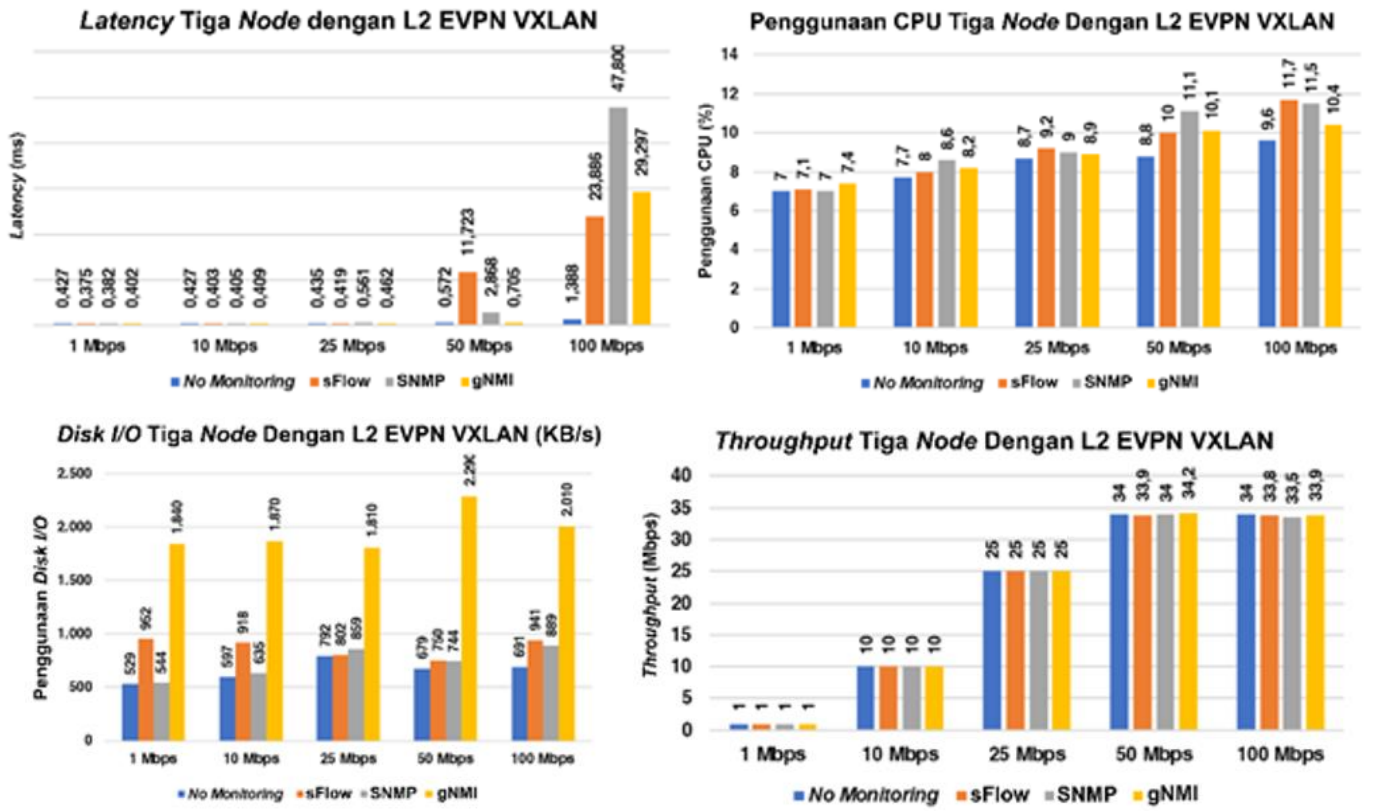
Subsistem gNMI menggunakan cara *pull* untuk mengubah konfigurasi pada perangkat jaringan dan menggunakan cara *push* (pengiriman informasi dari server ke perangkat pengumpul/*collector* tanpa permintaan) untuk memperoleh informasi status dan metrik dari perangkat tersebut. Subsistem gNMI perlu berlangganan pada *port* tertentu di perangkat jaringan agar dapat berkomunikasi dengan perangkat, yang dalam hal ini diatur pada *file* konfigurasi klien gNMI seperti pada Gambar 2.

Subsistem pemantauan SNMP menggunakan cara *polling* (secara berkala memeriksa atau mengirim permintaan kepada server untuk mendapatkan pembaruan) pada alamat *internet protocol* (IP) tertentu. Proses ini bertujuan untuk berkomunikasi dengan perangkat jaringan yang menanyakan status informasi sistem dan metrik dari perangkat jaringan tersebut. Perangkat membalasnya sesuai *object identifiers* (OID) yang diminta oleh pengumpul, dengan cara melakukan *poll* informasi tersebut kepada pengumpul.

Subsistem sFlow menggunakan cara yang berbeda dari subsistem lainnya. Pengumpul sFlow hanya perlu dikonfigurasi dengan alamat IP dari perangkat yang dituju. Kemudian, perangkat jaringan mengirim seluruh data yang ditemukan dalam protokol manajemen sFlow yang diatur oleh vendor. Pengumpul sFlow yang digunakan terintegrasi langsung dengan *time series database* (TSDB) Prometheus dan *dashboard* InMon dalam satu *image* Docker. Beberapa konfigurasi jaringan, konfigurasi *image* Docker, dan *environment variable* ditulis pada *file* konfigurasi *deployment* containerlab seperti yang ditunjukkan oleh arsitektur subsistem gNMI dalam Gambar 2.

D. KONFIGURASI SOCKET BUFFER

Perancangan sistem pemantauan jaringan dan topologi jaringan dalam penelitian ini dilakukan dengan containerlab.



Gambar 3. Data hasil pengujian dengan lalu lintas data UDP.

Containerlab menggunakan *bridge* Docker yang merupakan *driver* jaringan Docker secara *default*. *Driver* ini memungkinkan komunikasi antar-*container* berjalan di *host* Docker yang sama. *Bridge* ini membuat antarmuka jembatan virtual yang disebut "Docker0" pada sistem *host*, bertindak sebagai *virtual switch* untuk memfasilitasi komunikasi antar-*container*. *Bridge* Docker menggunakan variabel nilai *memory network buffer* atau *socket buffer* dari *host*. *Memory socket buffer* diatur dengan kapasitas *default* dan maksimum sebesar 32 MB, sehingga dapat menerima paket *jumbo size* (9.000 byte).

III. HASIL DAN PEMBAHASAN

Pengambilan data dalam penelitian ini dilakukan dengan cara mengamati secara langsung sumber daya komputasi yang digunakan ketika jaringan mengirimkan paket TCP atau *user datagram protocol* (UDP). *Tool* Glances digunakan dalam penelitian ini untuk menampilkan sumber daya pada *host* yang menjadi tempat simulasi jaringan *data center* dan pemantauan jaringan.

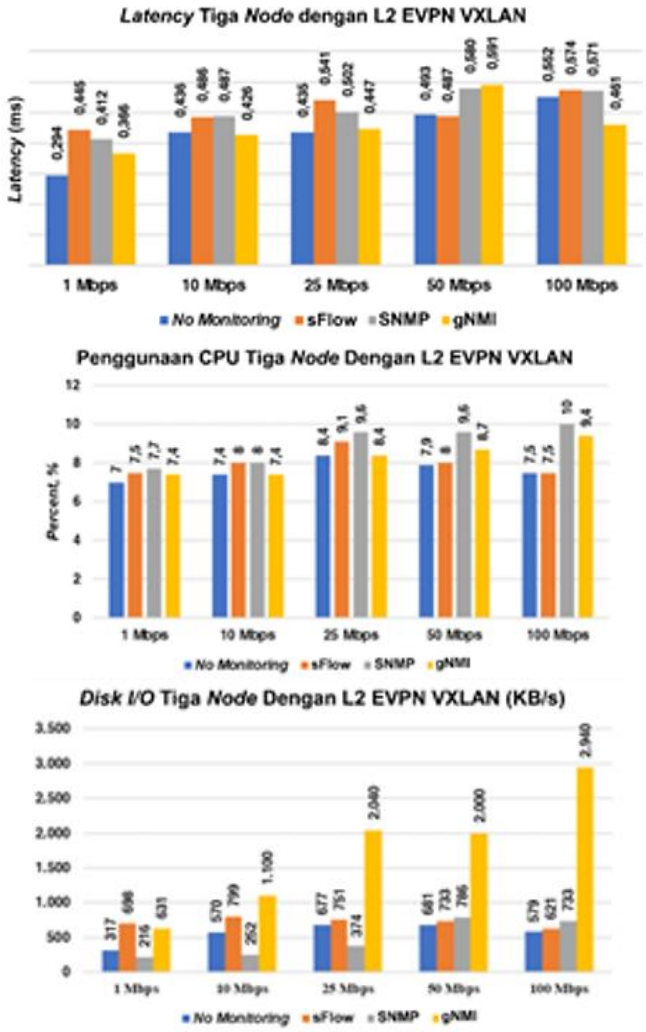
A. ANALISIS KINERJA SISTEM PEMANTAUAN BERBASIS gNMI

Hasil pengujian yang dilakukan dengan jenis paket UDP ditunjukkan pada Gambar 3. Grafik tersebut menunjukkan perbandingan ketiga sistem pemantauan dengan data metrik yang bersumber dari tiga *node* Nokia Service Router Linux (SR Linux) yang menjalankan protokol Layer 2 BGP EVPN VXLAN. Karena adanya batasan sistem operasi jaringan SR Linux berbasis Docker yang hanya sanggup mengolah 1.000 paket per detik, pengujian pada pesat data 50 Mbps dan 100 Mbps mengalami *packet loss* yang menyebabkan *throughput* menurun dan *latency* menjadi tinggi. Skenario ini masih sesuai tujuan, yaitu menganalisis kondisi jaringan, tetapi berubah dari skenario *elephant flow* menjadi skenario dengan kanal *lossy* dan *congestion*.

Hasil pengamatan pada parameter *throughput* dalam Gambar 3 tidak menunjukkan adanya perubahan signifikan dalam penggunaan *monitoring tool stack*. Namun, parameter *latency* pada Gambar 3 menunjukkan bahwa sistem pemantauan berbasis SNMP dan gNMI mengalami peningkatan *latency* pada kondisi jaringan *lossy* dan *congestion*. Utilitas CPU pada sistem pemantauan berbasis sFlow dan SNMP pada kondisi *lossy* dan *congestion* mengindikasikan kenaikan sekitar 2% dari topologi jaringan tanpa sistem pemantauan, Sementara itu, utilitas CPU sistem pemantauan dengan gNMI naik tidak lebih besar dari 1,5% pada kondisi jaringan *lossy* dan *congestion*.

Dalam pengamatan penggunaan RAM (5,24 GB tanpa sistem pemantauan), sistem *monitoring stack* berbasis sFlow menggunakan sumber daya yang lebih besar dibandingkan sistem pemantauan lainnya, yaitu hingga 6,24 GB (meningkat 1 GB). Di sisi lain, sistem pemantauan berbasis SNMP dan gNMI menggunakan sumber daya RAM kurang lebih 5,37 GB (meningkat sekitar 200 MB). Untuk parameter penggunaan *Disk* I/O, sistem pemantauan berbasis gNMI menggunakan sumber daya yang lebih besar dibandingkan protokol lainnya. Nilai yang ditunjukkan dalam pengamatan dapat mencapai lebih dari dua kali lipat dari *Disk* I/O sistem pemantauan lainnya. Pengujian dilanjutkan dengan lalu lintas bertipe paket TCP. Hasil pengujian ditunjukkan pada Gambar 4.

Pengamatan *latency* paket TCP di Gambar 4 menunjukkan adanya perbedaan dengan *latency* pada paket UDP. *Latency* TCP rata-rata berkisar antara 0,4 ms hingga 0,55 ms. Hasil pengujian antar sistem pemantauan tidak berbeda jauh jika dibandingkan dengan topologi tanpa sistem pemantauan. Hasil pengamatan pada penggunaan RAM dan penggunaan CPU dalam pengujian paket TCP ini tidak jauh berbeda hasilnya dengan pengujian paket UDP. Pada parameter penggunaan *Disk* I/O, hasilnya juga sama, yaitu sistem pemantauan berbasis



Gambar 4. Data hasil pengujian dengan lalu lintas data TCP.

gNMI menggunakan sumber daya dua kali lebih besar dibandingkan sistem pemantauan lainnya. Namun, nilai penggunaan sumber daya *Disk I/O* pada pengujian dengan paket data tipe TCP relatif lebih rendah dibandingkan pengujian dengan data UDP.

Pengambilan data dalam pengujian ini dilakukan secara manual dan penentuan nilai yang dicatat didasarkan pada nilai modus dalam waktu di antara interval *scraping* sistem pemantauan tersebut. Pengujian ini dilakukan dengan parameter durasi interval *scraping* yang sama, yaitu 15 detik. Sistem pemantauan SNMP dikonfigurasi untuk mengambil lima buah objek OID, sedangkan sFlow mengambil keseluruhan objek metrik yang disediakan oleh SR Linux. Sementara itu, sistem pemantauan berbasis gNMI dikonfigurasi untuk mengambil sembilan direktori objek telemetri yang berisikan 65 nilai objek telemetri. Dalam pengamatan saat pengujian, didapatkan hasil bahwa perilaku sistem dalam penggunaan CPU dapat naik 25% dari nilai yang dicantumkan dalam data untuk sistem pemantauan berbasis sFlow dan SNMP, sedangkan sistem pemantauan berbasis gNMI dapat meningkat hingga 50%.

B. ANALISIS PENGARUH DURASI INTERVAL PULLING DATA

Pada skenario pengujian ini, diinvestigasi pengaruh durasi interval *pulling* dari langganan telemetri gNMI terhadap penggunaan sumber daya dan kondisi jaringan yang aktif mengirim data. Interval Prometheus Agent yang ditetapkan dalam *file* konfigurasi perlu disinkronkan dengan durasi interval dari klien gNMI untuk mencegah ketidakakuratan informasi yang disimpan dalam TSDB.

TABEL II
DATA HASIL PENGAMATAN DURASI INTERVAL GNMI

Durasi Interval	CPU (%)	Memori (GByte)	*Disk I/O* (KB/s)	*Latency* (ms)
20 detik	8,5	5,31	1.780	0,460
15 detik	8,9	5,36	1.810	0,462
10 detik	11,7	5,36	2.450	0,489
5 detik	12,8	5,36	4.910	0,555
3 detik	13,4	5,38	6.920	0,544
1 detik	15,0	5,40	13.100	0,516

Tabel II menunjukkan pengaruh durasi interval *pulling* data pada telemetri gNMI. Penetapan durasi interval diawali dari 20 detik hingga interval yang mendekati *real-time*. Pengujian ini menggunakan jenis paket UDP dengan pesat data 25 Mbps dengan durasi pengujian 1 menit. Jumlah objek gNMI yang dilanggan sebanyak 65 objek. Pesat data 25 Mbps dipilih untuk mengetahui karakteristik sistem pemantauan berbasis gNMI pada kondisi ideal atau tidak mengalami *packet loss*.

Data hasil pengamatan pada Tabel II menunjukkan bahwa makin pendek interval *scraping* yang dikonfigurasi pada telemetri gNMI, makin besar tingkat penggunaan CPU pada *host*. Begitu juga pada tingkat penggunaan RAM, nilai penggunaan makin meningkat ketika interval *scraping* mendekati *real-time*. Hasil pengamatan pada penggunaan *Disk I/O* juga menunjukkan peningkatan nilai penggunaan ketika interval *scraping* mendekati nilai *real-time*. Nilai penggunaan *Disk I/O* meningkat hingga enam kali lipat dari skenario pengujian awal dengan interval *scraping* 20 detik. Parameter interval *scraping* ini patut dipertimbangkan ketika sistem pemantauan berbasis *streaming telemetry* gNMI akan digunakan pada lingkungan produksi. Hasil ini mungkin dapat berbeda ketika diimplementasikan pada sistem operasi jaringan lainnya.

C. ANALISIS PENGARUH JUMLAH OBJEK LANGGANAN DATA gNMI

Skenario pengujian ini bertujuan untuk mengetahui pengaruh jumlah objek langganan data parameter perangkat jaringan terhadap tingkat penggunaan dan lalu lintas jaringan komunikasi. Pengujian juga dilakukan dengan jenis paket UDP 25 mbps dengan durasi pengujian 1 menit. Interval *scraping* dikonfigurasi sebesar 3 detik. Tabel III menunjukkan hasil pengujian skenario ini. Jumlah objek langganan yang dimaksudkan dalam pengujian ini adalah objek terdalam yang terdapat dalam struktur hierarkis pohon YANG pada SR Linux.

Dari data hasil pengamatan pengujian yang ditunjukkan pada Tabel III, makin banyak objek gNMI yang dilanggan, makin bertambah juga tingkat penggunaan CPU. Begitu pula dengan tingkat penggunaan RAM pada *host*. Namun, nilai peningkatan penggunaan RAM tidak sebesar pada sistem pemantauan berbasis sFlow. Hasil pengamatan parameter utilitas *Disk* I/O juga menunjukkan peningkatan nilai seiring dengan bertambahnya jumlah objek gNMI yang dilanggan. Pengujian ini mengindikasikan bahwa selain durasi interval *scraping*, jumlah objek yang hendak diamati oleh sistem pemantauan juga perlu direncanakan dalam lingkungan produksi untuk mencegah degradasi kinerja sistem dan *host*.

TABEL III
DATA HASIL PENGAMATAN JUMLAH OBJEK LANGGANAN

Jumlah Objek Dilanggan	CPU (%)	Memori (GByte)	*Disk I/O* (KB/s)	*Latency* (ms)
21	10,0	5,31	4.420	0,420
45	10,7	5,33	5.390	0,483
65	13,6	5,38	6.090	0,474
97	20,4	5,40	10.400	0,590
132	21,9	5,40	19.500	0,578

TABEL IV
HASIL PENGAMATAN INTEGRASI *STREAMING TELEMETRY* GNMI DENGAN *LOG MONITORING*

Metrik	Telemetri gNMI	Telemetri gNMI + *Syslog*
Latency (ms)	0,462	0,465
Penggunaan CPU (%)	8,9	9,5
Memori (GByte)	5,36	5,43
Disk I/O (KB/s)	1.810	2.410

D. ANALISIS KINERJA INTEGRASI gNMI DENGAN Syslog

Pengujian pada skenario ini dilakukan menggunakan tipe paket UDP, dengan pesat data sebesar 25 mbps dalam durasi pengujian 2 menit dan interval *sampling* gNMI *telemetry* sebesar 15 detik. *Tool* yang digunakan untuk mendukung sistem *log monitoring stack* adalah Syslog-ng (v3.38.1), promtail (v2.7.4), dan grafana loki (v2.7.4). Pengujian ini mengacu pada lab yang dipaparkan oleh Nokia [23]. Hasil pengujian ditunjukkan pada Tabel IV.

Data pada Tabel IV menunjukkan bahwa integrasi *streaming telemetry* gNMI dengan Syslog tidak memengaruhi kondisi lalu lintas jaringan yang ditunjukkan dengan perubahan nilai *latency* yang tidak signifikan. Penggunaan RAM juga meningkat tidak terlalu signifikan, yaitu sebesar 70 MB. Namun, yang perlu diperhatikan adalah tingkat sumber daya *Disk I/O* yang meningkat sebesar 33% dari telemetri gNMI tanpa *log monitoring*.

Pengujian ini menggunakan tiga *network node*, interval *scraping* 15 detik, dan jumlah objek yang dilanggan sebanyak 65 objek. Jika kedua parameter tersebut diatur untuk menggunakan durasi interval lebih pendek dan mengonsumsi jumlah objek lebih banyak, *trade-off* yang telah ditunjukkan pada pengujian sebelumnya perlu diperhatikan. Pengimplementasian sistem pemantauan, khususnya berbasis *streaming telemetry* gNMI dalam lingkungan produksi, perlu memperhatikan dan merencanakan sumber daya *host* secara baik. Jika jumlah *node* lebih banyak, ada kemungkinan nilai penggunaan sumber daya *Disk I/O* juga ikut meningkat. Dalam sistem pemantauan terpusat atau terdistribusi, umumnya implementasi berjalan dalam satu *host* dengan implementasi aplikasi atau *microservice* lain. Jika konfigurasi atau parameter *trade-off streaming telemetry* gNMI tidak durencanakan dengan tepat, penggunaan sumber daya *Disk I/O* yang intensif dapat memengaruhi kinerja aplikasi atau *microservice* yang berjalan dalam *host* yang sama dengan sistem pemantauan.

IV. KESIMPULAN

Penelitian ini menunjukkan bahwa sistem pemantauan dengan protokol gNMI menggunakan CPU lebih rendah 0,5% dibanding protokol lainnya dengan paket UDP. Di sisi lain, ketika jenis paket yang digunakan adalah TCP, sistem berbasis SNMP dan gNMI sama-sama mengindikasikan kenaikan utilitas CPU. Untuk parameter penggunaan *Disk I/O*, sistem pemantauan berbasis gNMI menggunakan sumber daya secara lebih intensif dibandingkan protokol lainnya. Penelitian juga membahas integrasi *streaming telemetry* gNMI dan *log monitoring*, yang menunjukkan kenaikan penggunaan memori sebesar 70 MB dan kenaikan tingkat sumber daya *Disk I/O* sebesar 33%.

Dalam penelitian ini juga ditemukan indikasi peningkatan penggunaan CPU oleh sistem berbasis gNMI sebesar 50% dari data modus yang dicatat dalam pengamatan. Hasil analisis *trade-off* yang teridentifikasi dalam penelitian ini adalah jika interval *scraping* makin mendekati *real-time* dan objek yang dilanggan makin banyak, penggunaan *Disk I/O* dan CPU makin meningkat. Diharapkan penelitian berikutnya dilakukan pada jenis *node* yang homogen dan topologi *node* yang lebih kompleks serta dapat diuji secara praktik nyata pada perangkat keras yang mendukung protokol *streaming telemetry* gNMI.

KONFLIK KEPENTINGAN

Penulis dengan tegas menyatakan bahwa tidak ada konflik kepentingan yang mungkin memengaruhi hasil atau interpretasi penelitian ini. Penulis tidak memiliki kepentingan finansial, keuangan, atau pribadi yang dapat memengaruhi penilaian objektif terhadap data atau temuan penelitian. Penulis juga tidak memiliki keterlibatan dalam organisasi atau perusahaan yang dapat memengaruhi hasil penelitian ini.

KONTRIBUSI PENULIS

Konseptualisasi, Fierda Kurniacahya Ariefputra; metodologi, Fierda Kurniacahya Ariefputra dan Eueung Mulyana; perancangan sistem, Fierda Kurniacahya Ariefputra; pelaksana simulasi, Fierda Kurniacahya Ariefputra; pengambilan data, Fierda Kurniacahya Ariefputra; penulisan—penyusunan draf asli, Fierda Kurniacahya Ariefputra; peninjauan, Eueung Mulyana.

UCAPAN TERIMA KASIH

Terima kasih diucapkan kepada Roman Dodin selaku penemu dan pengembang Containerlab dan Nokia SR Linux yang membantu penulis mengatasi bug dalam penelitian.

REFERENSI

[1] B. Aggarwal, Q. Xiong, dan E. Schroeder-Butterfill, "Impact of the use of the internet on quality of life in older adults: Review of literature," *Prim. Health Care Res. Develop.*, vol. 21, hal. 1-6, Des. 2020, doi: 10.1017/S1463423620000584.

[2] J. Manyika, "Big data: The next frontier for innovation, competition, and productivity," 2011. [Online]. Tersedia: https://personal.utdallas.edu/~muratk/courses/cloud11f_files/MGI-full-report.pdf

[3] E. Permadi (2020) Industri cloud dan hosting di Indonesia, begini kondisi saat pandemi COVID-19. [Online], https://sumatra.bisnis.com/read/20200819/534/1280826/industri-cloud-dan-hosting-di-indonesia-begini-kondisi-saat-pandemi-covid-19, tanggal akses: 25-Feb-2023.

[4] E.B. Nadales (2023) Impact of traffic growth on networks and investment needs. [Online], https://www.telefonica.com/en/communication-room/blog/impact-of-traffic-growth-on-networks-and-investment-needs/, tanggal akses: 5-Jun-2023.

[5] M.Y.B. Rasyiidin dan F.A. Murad, "Monitoring server berbasis SNMP menggunakan Cacti pada server lokal," *J. Ilm. FIFO*, vol. 13, no. 1, hal. 14-23, Mei 2021, doi: 10.22441/fifo.2021.v13i1.002.

[6] A. Pradana, I.R. Widiasari, dan R. Efendi, "Implementasi sistem monitoring jaringan menggunakan Zabbix berbasis SNMP," *AITI*, vol. 19, no. 2, hal. 248–262, Nov. 2022, doi: 10.24246/aiti.v19i2.248-262.

[7] A. Sgambelluri dkk., "Reliable and scalable Kafka-based framework for optical network telemetry," *J. Opt. Commun. Netw.*, vol. 13, no. 10, hal. E42-E52, Okt 2021, doi: 10.1364/JOCN.424639.

[8] F. Paolucci, A. Sgambelluri, P. Castoldi, dan F. Cugini, "Telemetry solutions in disaggregated optical networks: An experimental view," *Opt.*

Fiber Commun. Conf. (OFC) 2021, 2021, hal. 1-3, doi: 10.1364/OFC.2021.W1G.1.

[9] R. Vilalta dkk., "Telemetry-enabled cloud-native transport SDN controller for real-time monitoring of optical transponders using gNMI," *2020 Eur. Conf. Opt. Commun. (ECOC)*, 2020, hal. 1-4, doi: 10.1109/ECOC48923.2020.9333143.

[10] A. Sgambelluri dkk., "Open source implementation of OpenConfig telemetry-enabled NETCONF agent," *2019 21st Int. Conf. Transparent Opt. Netw. (ICTON)*, 2019, hal. 1–4, doi: 10.1109/ICTON.2019.8840320.

[11] K.S. Mayer dkk., "Machine-learning-based soft-failure localization with partial software-defined networking telemetry," *J. Opt. Commun. Netw.*, vol. 13, no. 10, hal. E122-131, Okt 2021, doi: 10.1364/JOCN.424654.

[12] J.E. Simsarian dkk., "Demonstration of cloud-based streaming telemetry processing for optical network monitoring," *2021 Eur. Conf. Opt. Commun. (ECOC)*, 2021, hal. 1–4, doi: 10.1109/ECOC52684.2021.9605813.

[13] X. Cheng dkk., "IntStream: An intent-driven streaming network telemetry framework," *2021 17th Int. Conf. Netw. Service Manag. (CNSM)*, 2021, hal. 473–481, doi: 10.23919/CNSM52442.2021.9615520.

[14] R.A.K. Fezeu dan Z.-L. Zhang, "Anomalous model-driven-telemetry network-stream BGP detection," *2020 IEEE 28th Int. Conf. Netw. Protoc. (ICNP)*, 2020, hal. 1–6, doi: 10.1109/ICNP49622.2020.9259411.

[15] A. Sadasivarao dkk., "Demonstration of extensible threshold-based streaming telemetry for open DWDM analytics and verification," *Opt. Fiber Commun. Conf. (OFC) 2020*, 2020, hal. 1-3, doi: 10.1364/OFC.2020.M3Z.5.

[16] R.P. Pinto dkk., "Packet-optical differentiated survivability implemented by P4 slices and gNMI telemetry," *2023 Opt. Fiber Commun. Conf. Exhib. (OFC) 2023*, 2023, hal. 1-3, doi: 10.1364/OFC.2023.M1G.3.

[17] Ç. Kurt dan O.A. Erdem, "Real-time anomaly detection and mitigation using streaming telemetry in SDN," *Turkish J. Elect. Eng. Comput. Sci.*, vol. 28, no. 5, hal. 2448–2466, Sep. 2020, doi: 10.3906/elk-1909-112.

[18] I. Ivanov, "Comparing the performance of SNMP to network telemetry streaming with gRPC/GPB," *53rd Int. Sci. Conf. Inf. Commun. Energy Syst. Technol.*, 2018, hal. 175–178.

[19] E. Pettersson, "A comparison of pull-and push-based network monitoring solutions examining bandwidth and system resource usage," Skripsi, KTH Royal Institute of Technology, Stockholm, Swedia, 2021.

[20] B. Buresh dkk. *A Modern, Open, and Scalable Fabric VXLAN EVPN*. (2016). Tanggal akses: 7-Mar-2023. Tersedia: https://www.cisco.com/c/dam/en/us/td/docs/switches/datacenter/nexus9000/sw/vxlan_evpn/VXLAN_EVPN.pdf

[21] H. Song dkk. (2022) "Network Telemetry Framework RFC 9232," [Online], https://datatracker.ietf.org/doc/html/rfc9232, tanggal akses: 10-Mar-2023.

[22] M. Korshunov. Streaming telemetry: Considerations & challenges [Online]. Tersedia: https://ripe78.ripe.net/presentations/26-ripe78_Korshunov_Streaming_Telemetry_consideration_and_challenges_final.pdf

[23] R. Dodin, B. Claeys, dan M. Vahlenkamp, "Nokia SR Linux Streaming Telemetry Lab," [Online], https://github.com/srl-labs/srl-telemetry-lab, tanggal akses: 10-Mar-2023.